An Algorithm for Probabilistic, Totally-Ordered Temporal Projection

Drew McDermott

Research Report YALEU/DCS/RR-1014
March 1994

YALE UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

An Algorithm for Probabilistic, Totally-Ordered Temporal Projection

Drew McDermott

Research Report YALEU/DCS/RR-1014
March 1994

An Algorithm for Probabilistic, Totally-Ordered Temporal Projection

Drew McDermott*

March 3, 1994

Abstract

Temporal projection, defined as the prediction of what might happen when a plan is executed, is an important component of many planning algorithms. To achieve efficiency, it is desirable for a projection to be a totally ordered event sequence. To cope with uncertainty, the events must be generated using probabilistic rules. We require a rule language that allows us to specify what can happen when an event occurs, as well as what events can occur when certain propositions are true. The language has a formal semantics, which allows us to prove that a set of rules has a unique model (if it is "consistent"). This language supports a Monte Carlo style of projection, in which event sequences are sampled randomly using the probabilities in the rules. The output of the projector is a timeline that allows a planning algorithm to test the truth of propositions at arbitrary points. The algorithms for building and retrieving from the timeline can be shown to be correct. Experiments show that for a typical theory, the time to build a timeline is a quadratic function of the number of events in the timeline.

1 The Problem

Automated planners need to do temporal reasoning, that is, to decide what will be true at various times if their plans are executed, in support of planning operations (such as reordering plan steps) that depend on when various facts become true or false during plan execution. The main line of research in this area is to represent a plan as a partially ordered list of events (more precisely, as a totally ordered set of events with some of the orderings unknown), and to attempt to infer what must be true before or after each event. In many such efforts, it is assumed that the events' effects are all known and context-independent, so that the fact P is true after event e if and only if there is some event e' preceding or coinciding with e that has P as an effect, and such that for every event e'' with $\neg P$ as an effect, e'' precedes e' or follows e. (McAllester and Rosenblitt 1991, Dean and McDermott 1987, Chapman 1987, Allen 1984, VanBeek 1992)

*This work was supported by the Advanced Research Projects Agency of the Defense Department, under ONR Contract Number N00014-91-J-1577.

However, in general the effects of an event depend on what is true before the event, and in this more general setting the problem becomes much harder (Dean and Boddy 1988). One way to cope with the complexity is to stop trying to work with a partial ordering representing a set of possible total orderings. Instead, decouple the plan from the temporal database completely. Given a plan, we can generate several *projections* of the plan, each of which is a totally ordered possible execution scenario. Now, instead of asking, What must be true in all projections?, we can ask, What is true in the sample of projections generated so far? There are several advantages to this approach:

1. Plans can remain partially ordered. In fact, they can become arbitrary programs, so long as it is possible to predict how they might be executed at some level of detail.
2. The inference machinery for each projection can be fast and simple.
3. The more projections the planner generates, the more it learns about the current plan. Under time pressure, it can generate fewer projections.
4. Probabilistic world models can be handled.

In this paper, I describe an algorithm for probabilistic, totally-ordered database management. The planning algorithm this fits into is described in (McDermott 1992). (The temporal system described in that paper differs in detail from the version described here.) I will have nothing to say about the overall planning architecture. I will just assume that there is a projection module that takes a plan and produces a sequence of dated events. Each event occurs at a single time instant. (Events with duration are modeled as beginning at one instant and ending at another.) The job of the temporal inference system is to build a data structure called a *timeline* that records the events and their effects. It starts with a set of initial conditions, that is, facts true before the projection begins. At any point before or after the event sequence is complete, we must be able to give the system a query and a time point, and it will return a list of all instances of the query that are true at that point. For a given timeline, it always returns the same answer for a given query and time point, but exactly which answer is determined by the probabilities in the laws of physics. That is, if a new timeline were constructed from the same event sequence, the same query could give different results.

This is not the first work in the area of probabilistic projection. The formalism I develop here takes off from the work of Hanks (1990a,1990b,Hanks and McDermott 1994), who developed a theory of temporal representation, plus an algorithm that generated all possible timelines with their associated probabilities. The present work achieves more representational power at the cost of finding only approximations to the true probabilities of different outcomes. Other works, notably Kanazawa (Dean and Kanazawa 1989, Kanazawa 1992), have reduced probabilistic temporal reasoning to reasoning about Bayes nets (Pearl 1988) whose nodes represent the occurrence of different events and propositions at different times. Unfortunately, the resulting nets are not particularly manageable. The present work may be thought of as an application of stochastic simulation to a specialized Bayes net, extended to allow for the generation of spontaneous random events. However, my goal is not to evaluate probabilities accurately, but instead to generate plausible scenarios for plan execution. It may well be more valuable for plan debugging to be able to inspect samples of actual causal chains than to have an accurate assessment of probabilities.

There have been several other stabs at formalizing and carrying out probabilistic temporal reasoning. Thiébaux and Hertzberg (1992) have a formalism similar to mine (more expressive in some ways, less expressive in others), that allows them to treat plans as Markov processes. See also (Haddawy 1990).

2 An Idealized System

In a temporal setting, propositions are not merely true or false. They become true at a time point, persist for a while, become false at a time point, and so forth. I use the word *occasion* to denote a stretch of time over which a proposition is true.[1] When an occasion becomes false, it is said to be *clipped*. Each occasion corresponds to an atomic formula, such as `loc(box1, coords(5,6))`,[2] called the *proposition* of the occasion. Negation, variables, and such are not allowed as occasion propositions.

There are several entry points to the system:

1. **START-TIMELINE**: Creates a new timeline.
2. **TIMELINE-ADVANCE**: Adds new *exogenous* events to the timeline. They are called "exogenous" because from the point of view of the timeline manager, they are arbitrary and unmodeled. In the course of adding a new exogenous event, it may randomly spawn some *endogenous* events, that is, events whose occurrence is modeled by rules in the timeline manager's model.
3. **TIMELINE-RETRIEVE**: Used to infer what's true at the end of the timeline. Both **TIMELINE-ADVANCE** and **TIMELINE-RETRIEVE** use an internal routine called **TL-RETRIEVE**, which can determine what's true at an arbitrary point in the timeline.
4. **PAST-RETRIEVE**: Direct entry point to **TL-RETRIEVE**.

All these routines make and record random choices, based on the probabilities in the rules described below.

It is important to realize that in this paper I am speaking only of the temporal-reasoning system, not the overall planning system it is a part of (described inMcDermott 1992). Some of the ways the narrower system is described will seem parochical or warped. For example, the labels "exogenous" and "endogenous" applied to events seem backwards from the point of view of the overall planner. What I call "exogenous" events are those due to the planner's own actions, while "endogenous" events are the autonomous events that occur in the part of the world outside the planner's control. However, even though they are outside the agent's control, there is a probabilistic model of them, so the timeline system knows how to generate them; it has no model of the agent's actions.

Another possible point of confusion is that the timelines generated by the projection system are totally ordered, while the plans being projected are not. For example, our plan language, called RPL, allows for loops and parallel combinations of events. The plan `(N-TIMES K (PAR (P) (Q)))` specifies doing `P` and `Q` each `K` times, in no particular order on each iteration. `K` is a RPL variable, whose value can vary from execution to execution. A typical projection, with `K`=2, might look like

$$\texttt{Q.1.}A \rightarrow \texttt{Q.1.}B \rightarrow \texttt{P.1.}A \rightarrow \texttt{Q.1.}Z \rightarrow \texttt{P.1.}Z \rightarrow \texttt{P.2.}A \rightarrow \texttt{Q.2.}A \rightarrow \texttt{Q.2.}Z \rightarrow \texttt{P.2.}Z$$

[1] In Dean and McDermott 1987, these were called "time tokens." The implementation of occasions is not quite faithful to this formal definition, as we will see in Section 3.

[2] In the actual implementation, formulas and rules are expressed in a Lispish notation, which will be described in Section 3.

where X.i.Y is an event in the i'th execution of X. Events are labeled A, B, ..., Z. For instance, the first execution of Q has events Q.1.A →Q.1.B →Q.1.Z The events of P.1 are interleaved with the events of Q.1, and similarly for P.2 and Q.2. But although the interleavings differ from projection to projection, for any given projection the events are totally ordered. Finally, in this paper I do not explain how the action P generates events like P.1.A and P.1.Z. I'll just take events as given. Consult (McDermott 1992) for the linkage.

2.1 Probabilistic Rules

There are several forms of rule for expressing the "laws of physics" in a given domain:

- *precond*/*event* $\overset{prob}{|\!\!\rightarrow}$ *effect*: Whenever an event of the form *event* occurs, when *precond* is true, then, with probability *prob*, create and clip occasions as specified by *effect*. Used whenever an event is added. The *effect* can be a conjunction of atomic formulas and expressions of the form $\neg A$, where A is an atomic formula. The $\neg$ symbol in an *effect* is not interpreted as ordinary negation, but means that an occasion of A comes to an end with this event. We can read $\neg A$ as "clip A."
- *effect* $\overset{prob}{\leftarrow\!\!|}$ *event**precond*: Same meaning as $|\!\!\rightarrow$, but used to answer a query about *effect*.
- *precond* $\overset{elap}{\rightarrow\!\!|}$ *event*: Over any interval where *precond* is true, generates random endogenous events, "Poisson-distributed," with an average spacing of *elap* time units. (The exact nature of the distribution is explained below.)
- $p \leftarrow q$: If q is true at a time point, so is p. Used to answer queries about p. p is an atomic formula; q is a conjunction of literals.

Rules with connective $|\!\!\rightarrow$ or $\rightarrow\!\!|$ are called *forward-chaining rules.* Rule with connective $\leftarrow\!\!|$ or $\leftarrow$ are called *backward-chaining rules.* The *consequent* of a rule is the part on the same side of the connective as the arrowhead. The part on the other side (*precond* or q) is called the *antecedent.*

Rules and queries may contain variables, and the implementation uses unification to match queries with rules, but in the technical sections of this paper we will avoid thinking of the system as being a full first-order logic. Instead, I will treat each rule as a schema standing for all its ground instances. Because occasions cannot contain variables, we impose the requirement that if a variable occurs in the *consequent* of a forward-chaining rule, then it must also occur in the *antecedent* of that rule, or in its *event* if it has one. Similarly, if a variable occurs in the *antecedent* of a backward-chaining rule, it must also occur in the *consequent* or *event.* And if a variable occurs in the *prob* of a $|\!\!\rightarrow$ or the *elap* of a $\rightarrow\!\!|$, it must occur in the antecedent (or the event if it's a $|\!\!\rightarrow$); if it appears in the *prob* of a $\leftarrow\!\!|$, it must occur in the consequent.

In each case, the antecedent can be a conjunction of literals, that is, atomic formulas and expressions of the form $\neg A$, where A is atomic. The empty conjunction can be written **true**, or omitted. As I will show, this system is simple enough that $\neg$ can be implemented by the device of "negation as failure"

(Clark 1978). Because of this, I impose the restriction that any variable that occurs in a negated part of an antecedent must also occur in an unnegated part.

A rule instance is said to be *firable* at a time point if its precondition is true at that time point. It is said to *fire* if it is firable and its probabilistic test comes up true. ***The probabilistic test for a given firable rule instance is performed at most once at a given time point.*** (In the case of a _ ←| rule, if the query never occurs, the test is never performed.)

TIMELINE-ADVANCE uses _ →| rules to add endogenous events to the timeline. It calls **TL-RETRIEVE** to verify the preconditions of these rules, using _ ←| and _ ← _ rules. For every event added by **TIMELINE-ADVANCE**, |→_ rules are used to infer the consequences of the event.

Every timeline begins with an event of the form **START**. We can use this event to set up initial conditions,[3] as shown in the following example:

;*Initially,* **nugget1** *is at 10,0, and...*

$$\texttt{START} \overset{1}{\mid\!\!\rightarrow} \texttt{loc(nugget1, coords}(10,0))$$

;**T1000**, *an enemy robot, is at 0,0*

$$\texttt{START} \overset{1}{\mid\!\!\rightarrow} \texttt{loc(T1000, coords}(0,0))$$

;*Roughly every 10 seconds,* **T1000** *moves right.*

$$\overset{10}{\rightarrow\!\mid} \texttt{move(T1000}, 1, 0)$$

;*When an agent moves, its location changes.*

$$\texttt{loc}(b, \texttt{coords}(x,y))/\texttt{move}(b, \Delta x, \Delta y) \overset{1}{\mid\!\!\rightarrow} \texttt{loc}(b, \texttt{coords}(x+\Delta x, y+\Delta y)) \wedge \neg\texttt{loc}(b, \texttt{coords}(x,y))$$

;*(We don't really notate addition in rules this way;*
;*see Section 9.)*
;*Whenever* **T1000** *and the nugget are in the same place,*
;**T1000** *grasps the nugget, typically within 1 second.*

$$\texttt{loc(T1000, coords}(x,y)) \wedge \texttt{loc(nugget1, coords}(x,y)) \overset{1}{\rightarrow\!\mid} \texttt{grasp(T1000, nugget1)}$$

;*Grasping succeeds with a probability of 80%.*

$$\texttt{holding}(a,b) \overset{0.8}{\leftarrow\!\mid} \texttt{grasp}(a,b) \backslash \texttt{loc}(a, \texttt{coords}(x,y)) \wedge \texttt{loc}(b, \texttt{coords}(x,y))$$

This little model of the world might be useful if the planner had to predict the chances of preventing an enemy robot from stealing a gold nugget. The planner could project the plan by initializing the timeline, then using **TIMELINE-ADVANCE** to add the agent's own successive actions as exogenous events, then using **TIMELINE-RETRIEVE** with query $\texttt{loc(nugget1, coords}(10,0))$ to find out if the nugget is still at 10,0 at the end. The enemy will take about 100 seconds to get to 10,0, then repeatedly try to grasp the nugget, until it succeeds, when $\texttt{loc(nugget1, coords}(10,0))$ will be clipped and grasping will cease. Given these rules, the only way to stop the T1000 is to move it to a location where it will not reach the nugget.

[3] There are more flexible ways in the actual implementation, described in Section 3.

The only difference between $|\!\rightarrow\_$ and $\_\!\leftarrow|$ rules is when they are called. $\_\!\leftarrow|$ rules have the advantage that they are called only when the query they answer arises. I call the application of $\_\!\leftarrow|$ rules "backward chaining," and the application of $|\!\rightarrow\_$ rules "forward chaining" by analogy with their static counterparts. In the present example, there was really no reason to use a $\_\!\leftarrow|$ rule, because the $|\!\rightarrow\_$ rule is going to require querying **loc(nugget1, coords(...))** repeatedly anyway. This pattern, where the same query is repeated at many time points, distinguishes temporal inference from routine backward chaining, and leads to the optimizations described in Section 3.

2.2 Formal Semantics

So far my exposition of the meanings of the various types of rule has been informal. But below I will introduce some fairly intricate algorithms for making inferences using those rules, and we will want to verify that the algorithms work. Hence, I need to be more formal about the semantics of the rules.

Definition 1 A *world state* is a function from proposition symbols to $\{\#\mathtt{T}, \#\mathtt{F}, \bot\}$.

A world state s can be considered an assignment of truth values to every proposition symbol. I use **#T** and **#F** to denote boolean values; $\bot$ means "undefined," or "inconsistent." The state **F** assigns **#F** to every proposition symbol.

I will extend the meaning of states so that they apply to boolean combinations of propositions in the obvious way. In this paper, propositions will be notated as ground atomic formulas, but that notation is mainly a frill. All we require is a supply (possibly infinite) of symbols for propositions, which I will call $\mathcal{P}$. Similarly, we need a supply of event symbols, which I will call $\mathcal{Q}$.

Definition 2 An *occurrence* is a pair $\langle e, t\rangle$, where e is an event (type) and t is a particular *time*, a nonnegative real number. The *event* of the occurrence is e and the *date* is t. We will write such an occurrence as $e \downarrow t$.

Definition 3 An *occurrence sequence* is a sequence of occurrences, ordered by date. If C is a finite occurrence sequence, then we write its length as $\mathit{length}(C)$. If the dates in C are bounded, we write $\mathit{duration}(C)$ to mean the least upper bound of those dates.

Obviously, if C is finite, $\mathit{duration}(C) = \mathit{date}(C_{\mathit{length}(C)})$.

Definition 4 A *world of duration* L, where L is a real number > 0, is a complete history of a stretch of time of duration L. That is, it is a pair $\langle C, H\rangle$, where C is an occurrence sequence, such that the date of the last occurrence is $\leq L$, and H is a function from $[0, L]$ to world states. $H(0)$, the initial state, must be **F**, the state in which all proposition symbols $\mathcal{P}$ are false. If $t_1 < t_2$ and $H(t_1) \neq H(t_2)$ then there must be an occurrence $e \downarrow t \in C$ with $t_1 \leq t < t_2$.

If $W = (C, H)$ is a world of duration L and A is a proposition, and t is a date $\leq L$, I will write $(A \downarrow t)(W)$, read "A after t in W," to mean that there is some $\delta > 0$ such that for all t', $t < t' < t + \delta$, $H(t')(A)$. Similarly, I will write $(A \uparrow t)(W)$, read "A before t in W," to mean that there is some $\delta > 0$ such that for all t', $t - \delta < t' \leq t$, $H(t')(A)$. If c is an occurrence, I will write $A \uparrow c$ and $A \downarrow c$ to mean $A \uparrow \mathit{date}(c)$ and $A \downarrow \mathit{date}(c)$.

It should be obvious that in any world $W = (C, H)$ where c and d are consecutive occurrences $\in C$, and for any proposition A, $(A \uparrow d)(W) = (A \downarrow c)(W)$, and for all t, $\mathit{date}(c) < t < \mathit{date}(d)$, $(A \downarrow t)(W) = (A \uparrow t)(W) = (A \downarrow c)(W)$.

The next step is to attempt to define the idea of a model of a probabilistic theory.

Definition 5 *(Actually, an attempted definition)* If $\mathcal{T}$ is a set of rules as described in Section 2.1, $\mathcal{C}$ is an occurrence sequence of length n (the exogenous occurrences), and L is a real number $\geq \mathit{duration}(\mathcal{C})$, then an *$L$-model* of $\mathcal{T}$ and $\mathcal{C}$ is a pair $(\mathcal{W}, \mathcal{M})$, where $\mathcal{W}$ is a set of worlds of duration L such that for each $(C, H) \in \mathcal{W}$, $\mathcal{C} \subset C$; and $\mathcal{M}$ is a probability measure (Breiman 1969) on $\mathcal{W}$ that obeys certain restrictions, which I will now describe.

We can consider $A \uparrow t$, $A \downarrow t$, and $e \downarrow t$ to be boolean random variables, defined on the "outcome set" $\mathcal{W}$. As usual, we can combine random variables, letting $\mathcal{M}(\neg A) = 1 - \mathcal{M}(A)$, $\mathcal{M}(A|B) = \frac{\mathcal{M}(A \wedge B)}{\mathcal{M}(B)}$, etc. Define $\overline{A}$, the *annihilation of* A, where A is a conjunction, to be the conjunction of the negations of the conjuncts of A. (Example: $\overline{\mathtt{P} \wedge \neg \mathtt{Q}} = \neg \mathtt{P} \wedge \mathtt{Q}$.) For $(\mathcal{W}, \mathcal{M})$ to be a model, the measure $\mathcal{W}$ must be constrained to fit the rules $\mathcal{T}$ as follows:

1. *Initial blank slate:* For any proposition $A \in \mathcal{P}$, $\mathcal{M}(A \uparrow 0) = 0$.

2. *Event-effect rules when the events occur:* If $\mathcal{T}$ contains a rule instance $A/e \overset{r}{\longrightarrow} B$ or a rule instance $B \overset{r}{\longleftarrow} E \backslash A$, then for every date t, require that, for all nonempty conjunctions C of literals from B:

$$\mathcal{M}(C \downarrow t \mid E \downarrow t \wedge A \uparrow t \wedge \overline{B} \uparrow t) = r$$

3. *Static backward-chaining rules:* If $\mathcal{T}$ contains a rule instance $B \longleftarrow A$, then

$$\mathcal{M}(B \downarrow t \mid A \downarrow t) = 1$$

4. *Event-effect rules when the events don't occur:* Suppose B is an atomic formula, and let $R = \{R_i\}$ be the set of all instances of $\longrightarrow$ or $\longleftarrow$ rules whose consequents contain B or $\neg B$. If $R_i = A_i/E_i \overset{p_i}{\longrightarrow} C_i$ or $C_i \overset{p_i}{\longleftarrow} E_i \backslash A_i$, then let

$$D_i = A_i \wedge \overline{C_i}$$

Let S be the set of all the $\longleftarrow$ rules in whose consequents B occurs, and let

$$A = \bigvee_{r \in S} (\mathit{antecedent}(r))$$

(The D_i or A may be identically **false**, e.g., in the case where R or S is empty.) Then

$$\begin{aligned} \mathcal{M}(B\downarrow t \mid B\uparrow t \wedge N) &= 1 \\ \mathcal{M}(B\downarrow t \mid \neg B\uparrow t \wedge N) &= 0 \end{aligned}$$

where

$$\begin{aligned} N = \; & (\neg E_1\downarrow t \vee \neg D_1) \\ & \wedge(\neg E_2\downarrow t \vee \neg D_2) \\ & \wedge \ldots \\ & \wedge \neg A \end{aligned}$$

5. *Event-occurrence rules:* For every time point t such that no occurrence with date t is in C, and every event type E, let R be the set of all rule instances of the form $\ldots \xrightarrow{d_j} E$ in T, and suppose that S is an arbitrary nonempty subset of R, and let $A = \bigwedge_{R_j \in S} A_j$, and let $\lambda_S = \sum_{R_j \in S} 1/d_j$, where $R_j = A_j \xrightarrow{d_j} E$. Then if $\mathcal{M}(A\uparrow t) \neq 0$, require that

$$\mathcal{M}(\text{ some occurrence of } E \text{ between } t \text{ and } t + dt \mid A\uparrow t) = \lambda_S\, dt$$

If $N = \bigwedge_{R_j \in R} \neg A_j$, then

$$\mathcal{M}(\text{some occurrence of } E \text{ between } t \text{ and } t + dt \mid N\uparrow t) = 0$$

In most theories, R contains zero or one rule instance. In the former case, we require

$$\mathcal{M}(\text{ some occurrence of } E \text{ between } t \text{ and } t + dt) = 0$$

In the latter case, with just one instance $A \xrightarrow{d} e$, we require

$$\begin{aligned} \mathcal{M}(\text{ some occurrence of } E \text{ between } t \text{ and } t + dt \mid A\uparrow t) &= dt/d \\ \mathcal{M}(\text{ some occurrence of } E \text{ between } t \text{ and } t + dt \mid \neg A\uparrow t) &= 0 \end{aligned}$$

6. *Conditional independence:* If one of the previous clauses defines a conditional probability $\mathcal{M}(\alpha \mid \beta)$, which mention times t, then α is conditionally independent, given α, of all other random variables mentioning times on or before t. That is, for an arbitrary γ mentioning times on or before t, $\mathcal{M}(\alpha \mid \beta \wedge \gamma) = \mathcal{M}(\alpha \mid \beta)$.

Clause 5 of this definition is stated tersely but imprecisely. To make it rigorous, we must replace all statements of the form $\mathcal{M}(\text{ some occurrence of } e \text{ between } t \text{ and } t + dt \ldots) = x\, dt$ with

$$\lim_{\Delta t \to 0} \frac{\mathcal{M}(\text{ some occurrence of } e \text{ between } t \text{ and } t + \Delta t \ldots)}{\Delta t} = x$$

This quantity has the character of a *probability density*, and it is well defined only if $\mathcal{M}$ is defined over any small interval around almost every point t. There is a time point associated with every real number, so, if a theory contains $\xrightarrow{}$ rules, each of its models must have an uncountably infinite set of worlds.

In what follows, I will refer to this probability density with the slightly misleading notation $\mathcal{M}(e \downarrow t \ldots)$. So, even more tersely, we can state the constraint on a theory with one rule $A \overset{d}{\dashrightarrow} E$ governing occurrences of E as:

$$\begin{aligned} \mathcal{M}(E\downarrow t \mid A\uparrow t) &= 1/d \\ \mathcal{M}(E\downarrow t \mid \neg A\uparrow t) &= 0 \end{aligned}$$

In fact, I already made use of this notation in clauses 2 and 4 of Definition 5. The meaning of $\mathcal{M}(\ldots \mid c)$, where $c = E\downarrow t$, can only be rigorously specified using limits if $\mathcal{T}$ includes $\dashrightarrow$ rules.

Because of this infinity, we cannot assume that Definition 5 makes sense without putting in a little more work to show that there is exactly one $\mathcal{M}$ that satisfies it. The work is what you would expect: a limit process that defines $\mathcal{M}$ as the limit of a series of discrete approximations. The details are important but somewhat tortuous, so I have put them in an appendix. If you want to skip the appendix, you can just assume that Definition 5 defines a unique probability distribution.

Let me point out some consequences of Definition 5 that are fairly obvious even without the detailed analysis. Please attend to the role of negation in the clauses of Definition 5. Suppose we have a rule $e \overset{r}{\dashrightarrow} \neg A$. Then clause 2 says that $\mathcal{M}(\neg A\downarrow e \mid e \wedge A\uparrow e) = r$. Because the truth values of propositions can change only at occurrences, this formula means that any occasion of A that persists to an occurrence of e gets clipped with probability r.

Clause 4, with its negated annihilations, may sound daunting, so I will give an example. Suppose we have three rules

$$\begin{aligned} \texttt{P}\wedge\texttt{Q/squirt} &\overset{0.8}{\dashrightarrow} \texttt{B}\wedge\texttt{C} \\ \texttt{P/squirt} &\overset{0.6}{\dashrightarrow} \neg\texttt{B}\wedge\texttt{D} \\ \texttt{B} &\dashrightarrow \texttt{E} \end{aligned}$$

Then clause 4 states that

$$\begin{aligned} \mathcal{M}(B\downarrow t \mid B\uparrow t\wedge N) &= 1 \\ \mathcal{M}(B\downarrow t \mid \neg B\uparrow t\wedge N) &= 0 \end{aligned}$$

where N is defined as

$$\begin{aligned} &(\neg\texttt{squirt}\downarrow t \vee \neg\texttt{P}\uparrow t \vee \neg\texttt{Q}\uparrow t \vee \texttt{B}\uparrow t \vee \texttt{C}\uparrow t) \\ \wedge\ &(\neg\texttt{squirt}\downarrow t \vee \neg\texttt{P}\uparrow t \vee \neg\texttt{B}\uparrow t \vee \texttt{D}\uparrow t) \\ \wedge\ &\neg\texttt{E}\downarrow t \end{aligned}$$

One consequence of these rules is that if $\mathcal{T}$ contains two rule instances that specify different probabilities for their consequents in some worlds for an event sequence $\mathcal{C}$, then there is no model of $\mathcal{T}$ and $\mathcal{C}$. Such a theory, event-sequence pair is said to be *inconsistent*. A theory is *consistent* if it is consistent for all event sequences. For example, the theory

$$\begin{aligned} \texttt{START} &\overset{0.5}{\dashrightarrow} \texttt{A} \\ \texttt{A/bang} &\overset{0.8}{\dashrightarrow} \texttt{B}\wedge\texttt{C} \\ \texttt{A/bang} &\overset{0.6}{\dashrightarrow} \texttt{B}\wedge\texttt{D} \end{aligned}$$

is inconsistent for event sequence $\langle \mathbf{START} \downarrow 0, \mathbf{bang} \downarrow 1 \rangle$ because half the worlds in any model must have $\mathbf{A} \downarrow \mathbf{START}$, and those worlds assign two different probabilities to **B** after **bang**. For event sequences not containing **bang**, the theory is consistent. Observe the impact of assigning probability to every *subset* of the consequent of a rule.[4] If **A** is true before a **bang**, then the second rule states that **B**, **C**, and **B**∧**C** must all have probability 0.8 afterward. If we simply assigned a probability to **B**∧**C**, then the theory would be consistent after all, but the probabilities of the individual occasions would be underdetermined. For example, we could assign probabilities 0.6 to **BCD**, 0.2 to **BC**$\overline{\mathbf{D}}$, and 0.2 to $\mathbf{B}\overline{\mathbf{CD}}$, so that the probability of **B** alone was 1. Other assignments would give **B** different probabilities. My semantics avoids such indeterminacies.[5]

Compare the previous theory to this one:

$$\begin{array}{rcl} \mathbf{START} & \overset{0.5}{\mid\!\!\rightarrow\_} & \mathbf{A} \\ \mathbf{A/bang} & \overset{0.8}{\mid\!\!\rightarrow\_} & \mathbf{B} \\ \neg\mathbf{A/bang} & \overset{0.6}{\mid\!\!\rightarrow\_} & \mathbf{B} \end{array}$$

This theory is consistent for $\langle \mathbf{START} \downarrow 0, \mathbf{bang} \downarrow 1 \rangle$, because only one of the **B** rules is ever firable.

Another way for inconsistency to develop is for a $\_\leftarrow\_$ rule to disagree with a $\mid\!\!\rightarrow\_$ rule. However, the following theory is *consistent:*

$$\begin{array}{rcl} \mathbf{bang} & \overset{0.8}{\mid\!\!\rightarrow\_} & \mathbf{A} \\ \mathbf{bang} & \overset{0.6}{\mid\!\!\rightarrow\_} & \neg\mathbf{A} \end{array}$$

If a sequence of **bang**s occurs, then there are worlds where both rules fire. However, the first rule causes occasions of **A** to be created, and the second causes them to be clipped. Hence the two rules can never fire at the same eventinstant.

2.3 An Abstract Algorithm

In a later section, I will discuss the actual implementation of the algorithm. Here I will describe an idealized version that makes it clear how the algorithm works. One of the idealizations I will make is to ignore $\_\leftarrow\_$ rules, which clutter the exposition. I will return to them in Section 3. An idealization I will *not* make is to ignore $\_\leftarrow\mid$ rules. If we got rid of backward chaining, then **TL-RETRIEVE** would never have to look before the time point it had been asked about, and **TIMELINE-ADVANCE** would do all the inferential work. However, that would require us to initialize the timeline with every proposition the system believes. As I will discuss in greater depth in Section 3.1, that's an awkward requirement to meet. Hence we must adopt a style of "lazy projection" in which points along the timeline, especially the initial point, are constructed incrementally as queries are posed about them. (Cf. Hanks 1990a.)

[4] An alternative way to state the condition is to take the consequent of a rule, and generate all possible combinations of "signs" for the atomic formulas in it. Every combination except the one given must have probability zero. For the case at hand, the second rule would assign conditional probability zero to $\mathbf{B} \wedge \neg\mathbf{C}$, $\neg\mathbf{B} \wedge \mathbf{C}$, and $\neg\mathbf{B} \wedge \neg\mathbf{C}$.

[5] Of course, by making additional assumptions, it is possible to assign probabilities in cases like these. One such approach is explored by Thiébaux and Hertzberg (1992). In my experience, it is usually preferable to rewrite rules to avoid inconsistencies.

A timeline keeps track of events and the occasions that they delimit. The events are totally ordered, and no proposition changes in truth value between two adjacent events. Hence we can implement timelines, essentially, as lists of *eventinstants*, defined thus. An eventinstant is a record that contains:

- A date: The time of occurrence of the eventinstant, relative to the beginning of the timeline.
- Happenings: The events that occur here (zero or one).
- Overlapping occasions: Occasions that are true here (that is, they overlap this eventinstant, either because they begin here, or because they begin before and are not clipped here or before).
- Clipped ocasions: A list of occasions that were clipped at this eventinstant.
- Established queries: A list of atomic queries that have been tested at this eventinstant.
- Clipnotes: A table of ⟨ atomic-formula, boolean ⟩ pairs, where the boolean records whether the decision was to clip or not to clip the atomic formula at this eveninstant

The actual structure of an eventinstant is more complex, but this will do for now.

Each occasion is a record that contains:

- A proposition: a ground atomic formula
- Begin: The eventinstant where the occasion begins to be true
- End: The eventinstant where the occasion ceases to be true, or **#F** if that eventinstant is not yet known.

The fact that we store a list of occasions, each associated with an atomic formula, means that we are making the *closed-world assumption* for occasions: if no occasion for proposition A overlaps a time instant, then the algorithm will take A to be false at that time instant.

The established-queries list and the clipnotes table of an eventinstant play a crucial role. Once a query has been processed at an eventinstant, the inferred instances must be stored in this list. Such a query is said to have been *established at* this eventinstant. There are two good reasons for saving this information. First, it serves as a cache. When backward-chaining for answers to a query, the system must check previous eventinstants to see if the query can be infered there and shown to persist to the point of interest. If a query is repeated at every time point, which is a common occurrence in our application, this check will propagate back to previous eventinstants many times. The cache saves repeating rule application. Second, and even more important, we simply cannot run a probabilistic rule more than once at a given eventinstant. Once the answer to a query has been randomly selected, we must record the outcome, and avoid doing another random selection later.

A timeline is then essentially a list of eventinstants. In the Lisp implementation, the list is kept in reverse chronological order, so that adding to the timeline requires **CONS**ing a new eventinstant on, and backward chaining requires **CDR**ing through the list.

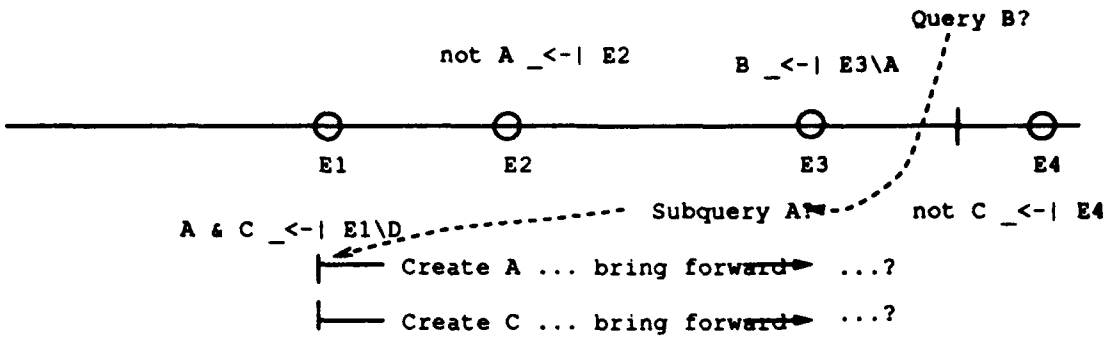


Figure 1: Interleaving Backward Chaining and Occasion Creation

I will use the term *time point* to refer to a position in this list. The *initial* time point corresponds to the beginning of the list. Every noninitial time point is associated with an eventinstant, namely, the one that just occurred (the **CAR** of the list in the implementation). Because of this near one-to-one correspondence between time points and eventinstants, I will refer to the date, happenings, etc. of a noninitial time point, meaning the date, happenings, etc. of the associated eventinstant. The *past* of a noninitial time point is the time point that is its immediate predecessor (its **CDR** in the implementation). Because no occasion changes in truth value between eventinstants, we do not really have continuous time, but in reasoning about endogenous events the system will have to choose real-valued dates at which to place the next event. In describing this part of the system, I will use the term *time instant* to describe one of the uncountable number of anonymous points between two eventinstants.

TL-RETRIEVE is the basic entry point to the temporal-inference system, so I will start by describing it. It takes as argument a query, a time point, and a timeline. It returns two values: a list of substitutions that answer the query (possibly empty), and a list of new occasions created while answering it (usually empty). This second value is quite important. Even though a query may have been asked at a point late in a timeline, answering it may require running a _ ←| rule at a point arbitrarily early in the timeline. An important special case is rules that trigger off **START**.[6] When a rule fires, it creates new occasions, which persist for a while, but may get clipped eventually. Getting **TL-RETRIEVE** right requires synchronizing this activity with backward chaining. When new occasions are created, the system will be in the midst of exploring answers to a query at a future time point. See Figure 1. It will be several layers down in a recursion involving possibly a cascade of _ ←| rules. If the system tried to figure out the lifetime of an occasion as soon as it was created, it would end up starting several new backward-chaining computations in the same place in the timeline as the existing computation, and inconsistent answers and infinite recursion would ensue. On the other hand, if it postponed figuring out the lifetime until the dust had settled on the current computation, then the current computation might not see the correct lifetime when it resumed at later time points.

The only solution is to interleave backward chaining with creation of new occasions. When **TL-RETRIEVE** returns, it guarantees that the timeline is in a consistent state up through the given time point, and that the new occasions it is returning persist that far. If it was called recursively, its caller must check whether the occasions persist to the next time point. If they do, they get returned up the line to the next

[6] and, in the actual implementation, timeline initializers, described in Section 3.

caller; otherwise, they get clipped. This persistence check is calling "bringing the occasions forward."

I will present algorithms in a pseudo-code style with vaguely Algol-like syntax. Italics will be used for comments, and for parts of the code that are not worth going into detail about. I will notate multiple-value return with angle brackets. If a function F returns an expression like $<x_1, x_2, \ldots>$, then its caller can capture those values by saying

```
Let <v1, v2, ...> = F(...)
  —body-of-Let—
```

As this example shows, scope is indicated with indentation wherever possible. To indicate random binary choices, I write a call to a procedure `random-choice(`*prob*`)`.

Here is the code:

```
TL-RETRIEVE(query,timepoint,timeline)
  If query is a conjunction,
    then call TL-RETRIEVE on each conjunct and combine the results,
         ordering the conjuncts so "¬" conjuncts occur last.
  Else if timepoint is the initial timepoint of timeline
    then <if query=true or ¬A for some A
            then (empty substitution)
            else (),
          ()>

  Else if query is of form ¬Q,
    then Let < anses, newoccs > = TL-RETRIEVE(Q,timepoint,timeline)
           <if anses = () then (empty substitution)
                          else (),
            newoccs>
    else Let newoccs = Maybe-establish-query(query,timepoint,timeline)
           <Match-against-occasions(query,timepoint),
            newoccs>

Match-against-occasions(query,timepoint)
  Match query against all occasions in overlapping-occasions(timepoint)
  Return a list of substitutions, one for each successful match

Maybe-establish-query(query,timepoint,timeline)
```

```
  If query is a variant of some member of established-queries(timepoint)
    then ()
    else Establish-query(query,timepoint,timeline)

Establish-query(query,timepoint,timeline)
  If timepoint is initial time point of timeline
    then ()
    else Let from-past =
              Bring-forward(Maybe-establish-query(query,past(timepoint),timeline),
                            timepoint,timeline)
         from-past ∪ Establish-after(query,timepoint,timeline)

Establish-after(query,timepoint,timeline)
  Let newoccs = '()
    For each rule Q1 ∧ Q2 ∧ ... ∧ Qn -p-| e\C,
              where query unifies with a Qi and event of timepoint unifies with e
                    with unifier θ
      newoccs := newoccs ∪ Try-back-chain-rule(rule,θ,timepoint,timeline);
    add query to established-queries(timepoint);
    newoccs

Try-back-chain-rule(rule,substitution,timepoint,timeline)
  ;rule is of form Q1 ∧ Q2 ∧ ... ∧ Qn -p-| e\C
  Let precond = Apply substitution to C
    Let <anses,occs> = TL-RETRIEVE(precond ∧ ¬Q1 ∧ ... ∧ ¬Qn,
                                   past(timepoint),timeline)
      Let newoccs = Bring-forward(occs,timepoint,timeline)
        For each ρ in anses
          newoccs := newoccs
                     ∪ Conclude-from-rule(rule,ρ,timepoint,timeline);
        Note that query is established at timepoint;
        newoccs

Conclude-from-rule(rule,substitution,timepoint,timeline)
  ;rule is of form Q1 ∧ Q2 ∧ ... ∧ Qn -p-| e\C
```

```
;              or C/e |-p-> Q1 ∧ Q2 ∧ ... ∧ Qn
Let D1 ∧ D2 ∧ ... ∧ Dn = Apply substitution to Q1 ∧ Q2 ∧ ... ∧ Qn
and prob = Apply substitution to p
and newoccs = '()
  If random-choice(prob) ;(rule fires)
     then For each Di
            If of form ¬A
              then if no entry ⟨A, ...⟩ ∈ clipnotes(timepoint)
                       ;No previous decision has been made whether
                       ;A is clipped at timepoint
                       then add ⟨A, #T⟩ to clipnotes(timepoint)
            Else if Di ∉ established-queries(timepoint)
              then ;Note that Di is established at timepoint
                   add Di to established-queries(timepoint);
                   Let occ = Create new occasion with proposition=Di
                                                  and begin=timepoint
                                                  and end=#F
                     add occ to overlapping-occasions(timepoint);
                     newoccs := cons(occ,newoccs)
     else For each Di
            If of form ¬A
              then if no entry ⟨A, ...⟩ ∈ clipnotes(timepoint)
                       ;No previous decision has been made whether
                       ;A is clipped at timepoint
                       then add ⟨A, #F⟩ to clipnotes(timepoint)
              else ;Note that Di has been established at timepoint
                   add Di to established-queries(timepoint);
  newoccs
```

Notice how the **established-queries** and **clipnotes** tables are used by these routines. The basic idea is that, for a given time point, no query is processed more than once, and no decision to create or clip an occasion is made more than once. Every time one of these things happens, it is recorded in the appropriate table to prevent its happening again. It would be possible to suppress the established-queries manipulations in **Maybe-establish-query** and **Establish-after**, although it would be less efficient. It is not possible to avoid the establish-queries manipulations in **Conclude-from-rule**, because consideration of the same literals can arise from consideration of different queries. For example, the rule

$\mathtt{B}(x,y) \overset{0.8}{\text{-}\!\leftarrow\!|} \mathtt{E}\backslash\mathtt{A}(x,y)$ can lead to consideration of the literal $\mathtt{B}(\mathtt{a},\mathtt{b})$ from both the query $\mathtt{B}(\mathtt{a},y)$ and the query $\mathtt{B}(x,\mathtt{b})$.[7]

Unlike the established-queries table, the clipnotes table records the outcome of the clipping decision as well as the fact that it was made. As we will see when considering $|\!\rightarrow\!\text{-}$ rules, when the system notices whether a proposition is clipped at a time point, it may or may not know whether an occasion with that proposition overlaps that time point. So the algorithm must save the clipping-decision result for future reference. **Bring-forward** then retrieves the outcome from the table to see what is clipped. Here's the code:

```
; This procedure is responsible for bringing new occasions forward
; and checking whether they survive through timepoint
Bring-forward(occasions,timepoint,timeline)
  Let to-future = () ; The list of survivors we're going to asemble
    For each occ ∈ occasions
      Let <clipped,newoccs> = Been-clipped(occ,timepoint,timeline)
        to-future := to-future ∪ newoccs;
        If clipped
           then end(occ) := timepoint
           else (Add occ to overlapping-occasions(timepoint);
                 to-future := cons(occ,to-future));
    to-future

; Test for whether occasion gets clipped at timepoint
; Return two values: boolean indicating whether it got clipped,
;plus (subtle requirement) new occasions created when deciding
; whether this one is clipped(!)
Been-clipped (occasion,timepoint,timeline)
  Let prop = proposition(occasion)
  and newoccs = '()
    For each rule of the form Q1 ∧ Q2 ∧ ... ∧ Qn -p-| e\C
                where some Qi is of form ¬P,
                      and prop unifies with P
                      and event of timepoint unifies with e
                          with unifier θ
      newoccs := newoccs ∪ Try-back-chain-rule(rule,θ,timepoint,timeline);
```

[7] An alternative idea would be to keep a "used rule instance" table in addition to the established-queries table.

```
<if There is an entry (prop,c) in clipnotes(timepoint),
    then c
    else #F,
newoccs>
```

The following lemma is obvious but useful:

Lemma 1 `Conclude-from-rule` makes a decision about creating or clipping an occasion at a time point only if the decision has not been made before.

Proof: Each such decision is recorded in the `established-queries` or `clipnotes` table for the time point, and blocks repetitions. QED

To prove that `TL-RETRIEVE` works, I first state two definitions.

Definition 6 A timeline is *properly established through time point* t if and only if t is initial or for every time point p before or coinciding with t, and every occasion $n = \langle A, c_1, c_2 \rangle$ in any eventinstant's `overlapping-occasions` list, $n \in$ `overlapping-occasions`(p) if and only if $date(c_1) \leq date(p) < date(c_2)$, taking any date to be $< date(c_2)$ when $date(c_2) = \#\mathbf{F}$.

Definition 7 A timeline is *properly clipped through time point* t if for every time point p before or coinciding with t, except the initial one, if $c \in$ `overlapping-occasions(past(`p`))`, where c has proposition A, then

- there is an entry $\langle A, \#\mathbf{T} \rangle \in$ `clipnotes`(p) if and only if $c \notin$ `overlapping-occasions`(p);
- if there is no entry for A in `clipnotes`(p) then there is no $|\rightarrow \_$ or $\_ \leftarrow|$ rule whose consequent includes a literal $\neg A'$, where A' unifies with A.

Now to state that `TL-RETRIEVE` works correctly:

Lemma 2 If a timeline L for a consistent theory $\mathcal{T}$ is properly established and properly clipped through time point t, then `TL-RETRIEVE`(q, t, L) will

1. make a decision about creating or clipping an occasion at a time point only if the decision has not been made before;
2. make and record decisions about occasions whose propositions are subsumed by atomic formulas occuring in q, such that the truth value of all instances of q at t will be decided and recorded;
3. make all decisions based on the correct conditional probabilities, as laid out in Definition 5;
4. leave L properly established and properly clipped through t;

5. return all answers that follow from the decisions made, plus a list of all occasions created that overlap t, each of which will be of the form $(A, c, \#\mathbf{F})$.

Proof: The first clause follows from the fact that **Conclude-from-rule** is the only routine that creates occasions, and **Bring-forward** is the only one that clips them. Both respect the decisions taken by **Conclude-from-rule**, so clause 1 follows by Lemma 1.

Clause 2 will follow if we can show that it holds for any atomic query, because **TL-RETRIEVE** breaks a compound query down into atomic pieces.

Clauses 2–5 are proven by induction on the number of eventinstants up through t, taking the query to be atomic as just mentioned. If t is the initial time point of L, then L will vacuously remain properly clipped and established through t. No probabilistic decisions will be made or recorded, and only formulas of the form **true** or $\neg A$ will be taken as true. No occasions will be created or returned, and the empty substitution or nothing will be returned.

Now assume clauses 2–5 of the theorem are true up to t, and consider t's successor t' in L.

Suppose that the algorithm is about to make a decision, in **Conclude-from-rule**, about whether to clip or create an occasion n with proposition A. **Conclude-from-rule** is called in **Try-back-chain-rule**, after a call to **TL-RETRIEVE** with

$$\textbf{query} = \textit{precondition of rule} \wedge \overline{\textit{consequent of rule}}$$

and **time point** $t =$ **past**(*current time point*). This query must have succeeded, thus finding a firable rule instance whose consequent contains A or $\neg A$. (Because T is consistent, there will be just one such rule instance.) By the induction hypothesis, the recursive call to **TL-RETRIEVE** will make enough decisions about this subquery to fix the truth values of all its instances at t, and will return the list P of all newly created occasions that persist through t. In particular, to get to this point in **Conclude-from-rule**, it must have found the substitution corresponding to A. Hence when it makes a random choice, it uses the probability from the rule instance obtained using this substitution, so that the chance of creating or clipping A is as described in clause 2 of Definition 5.

Now consider an occasion of A in **overlapping-occasions**(t), such that no rule involving A or $\neg A$ ever becomes firable at t' as a result of decisions made by recursive calls to **TL-RETRIEVE**. Because L was properly established and clipped through t' when we started, and, by induction, it is properly clipped through t after all recursive calls to **TL-RETRIEVE**, then either $n \in$**overlapping-occasions**(t') or $n \in$ some new-occasions list returned by a recursive call to **TL-RETRIEVE**. In the latter case, **Bring-forward** will have been called on the new-occasions list, and, by the induction hypothesis, decisions will have been made that fix at **false** the truth value of the formula corresponding to D in clause 4 of Definition 5. Hence **Bring-forward** will add n to the overlapping-occasions list of t'.

To prove clause 5, notice that the occasions returned by **TL-RETRIEVE** come from two sources: those created in **Conclude-from-rule** and those returned by recursive calls to **TL-RETRIEVE**. The former category all get returned, and all have **end=#F**. The latter category also have **end=#F** (by induction hypothesis), but they get passed through **Bring-forward**, which either clips them and resets their **ends**, or returns them, still with **end=#F**.

Finally, we need to show that the timeline remains properly established and clipped through t'. Let n be an occasion $\in$**overlapping-occasions**(t). If n was present before this call to **TL-RETRIEVE**, then its status in **overlapping-occasions**(t') and **clipnotes**(t') will not be changed. So suppose that n is a newly created occasion of A. Because it is new, and doesn't start at t', it will have been returned by some recursive call to **TL-RETRIEVE**. It will then be checked by **Been-clipped**, and will be omitted from **overlapping-occasions**(t') if only if an entry $\langle A, \#\mathbf{T}\rangle$ is added to the clipnotes for t'. Hence the timeline will remain properly clipped. To show it remains properly established, we also need to consider occasions that begin at t'. But these are always put in **overlapping-occasions**(t'). QED

The other main entry point to the system is **TIMELINE-ADVANCE**, which adds a single new eventinstant to a timeline. The easy part of its job is to run $|\!\rightarrow\_$ rules for the new eventinstant. The tricky part is to run $\_\rightarrow|$ rules. Recall that the meaning of $A \overset{d}{\_\rightarrow|} e$ is that over any tiny interval (duration "dt") where A is true, events of type e tend to occur with probability dt/d. What this entails is that, over an interval of length l, the probability that an e does not occur is $\exp(-\lambda l)$, where $\lambda = 1/d$ (Breiman 1969,Feller 1970).[8] In simple cases, such a rule should generate a random series of e events, according to a Poisson distribution. But every time an event occurs, $|\!\rightarrow\_$ rules can change the set of true propositions. Consider these rules:

$$\mathbf{A} \quad \overset{10}{\_\rightarrow|} \quad \mathbf{E}$$
$$\mathbf{E} \quad |\!\overset{1}{\rightarrow}\_ \quad \neg\mathbf{A}$$

the first rule suggests that any occasion of **A** is punctuated by random occurrences of **E** every 10 time units (on the average). But the second makes it clear that in fact the first such occurrence will be the last.

The best way to think of the meaning of $\_\rightarrow|$ rules is this: At any time instant, there is a set of firable $\_\rightarrow|$ rules. Each is like a ticking time bomb, with random elapsed times until they fire or until the next exogenous event occurs. If one of the rules fires, a new event is created, the set of firable rule instances gets recomputed, and the process resumes. In a sense, this process never stops, but the system does not bother to model past the last exogenous event. Hence the job of **TIMELINE-ADVANCE** is to tack on a new exogenous eventinstant, then fill in with endogenous events the time interval between the eventinstant's predecessor and the new eventinstant. I will call this the *time-passage interval.*

It is a well known fact (see, e.g., Breiman 1969) that a set of independent processes, each of which behaves as described by a $\_\rightarrow|$ rule instance with frequency λ_i, can be modeled as a single process with $\lambda = \sum_i \lambda_i$. Hence the chance that no endogenous event occurs in a time-passage interval of length l is $\exp(-\sum_i \lambda_i)l$. If an event does occur, then the chance that it is of the type belonging to process i is λ_i/λ.

I will now present the algorithm. I use the notation **date**(L) to denote the date of the last eventinstant in timeline L.

;Add a new eventinstant at the end of **timeline**, *after* **elapsed**
;time units. Put an event of type **event** *at the new eventinstant.*

[8] The only reason to use d instead of λ in the rule is that in the implementation, it is occasionally useful to set $d = 0$.

```
TIMELINE-ADVANCE (event,elapsed,timeline)
   Let advdate = date(timeline) + elapsed
     Pass-time(timeline,advdate);
     Add-eventinstant(advdate,timeline);
     Add-event(event,timeline)

;This routine fills in a time-passage interval with endogenous events
Pass-time(timeline,nextdate)
   Let rl = Event-rule-instances(timeline)
     If not empty(rl)
        then Let λ = ∑(λ_i) for every rule instance in rl
               Let prob = 1 − e^{−λ(nextdate−date(timeline))}
                 if random-choice(prob)
                    then Let x = random real number between 0 and 1
                           Let when = −log(1−x×prob) / λ
                             Let which = pick rule i from rl
                                         with probability λ_i/λ
                               One-endogenous-event(which,timeline,
                                                    date(timeline)+when,
                                                    nextdate)

Event-rule-instances(timeline)
  Let instances=()
    For every rule A −d→| E
      Let <anses,newoccs> = TL-RETRIEVE(A,last-time-point(timeline),timeline)
        Ignore newoccs — they'll be picked up by the call to Check-clipping;
        For a in anses
          add instance of rule for a to instances;
   instances

;Add event from rule-instance to timeline at time newdate,
;then pass time until thendate
One-endogenous-event(rule-instance,timeline,newdate,thendate)
  Add-eventinstant(newdate,timeline);
  Add-event(event from rule-instance,timeline);
  Pass-time(timeline,thendate)
```

```
Add-eventinstant(date,timeline)
   Add one more eventinstant, with date date, to the list in timeline

Add-event(event,timeline)
   Add event to happenings(last eventinstant of timeline);
   Forward-chain(event,timeline);
   Check-clipping(timeline)

Forward-chain(event,timeline)
  For every rule A/e |-P-> B, where e unifies with event with substitution θ
    Let <anses,newoccs>
            = TL-RETRIEVE(θA,penultimate-time-point(timeline),timeline)
      Ignore newoccs (see comment in Event-rule-instances);
      For a in anses
        Conclude-from-rule(rule,a,last-time-point(timeline),timeline)

Check-clipping(timeline)
  For each occ ∈ overlapping-occasions(penultimate-time-point(timeline))
    If Been-clipped(occ,last-time-point(timeline),timeline)
      then end(occ) := timepoint
      else add occ to overlapping-occasions(last-time-point(timeline))
```

This algorithm is somewhat unusual in that its termination condition is probabilistic. It can in principle loop, interpolating new events, forever, as **Pass-time** calls **One-endogenous-event** and **One-endogenous-event** calls **Pass-time**. Of course, the probability of this happening is zero.

Lemma 3 If a timeline L for a consistent theory $\mathcal{T}$ is properly ended, and properly established and properly clipped through its last time point, then **TIMELINE-ADVANCE**(e, l, L) will

1. make a decision about creating or clipping an occasion at a time point only if the decision has not been made before;
2. make all decisions based on the correct conditional probabilities, as laid out in Definition 5;
3. leave L properly established and properly clipped through t.

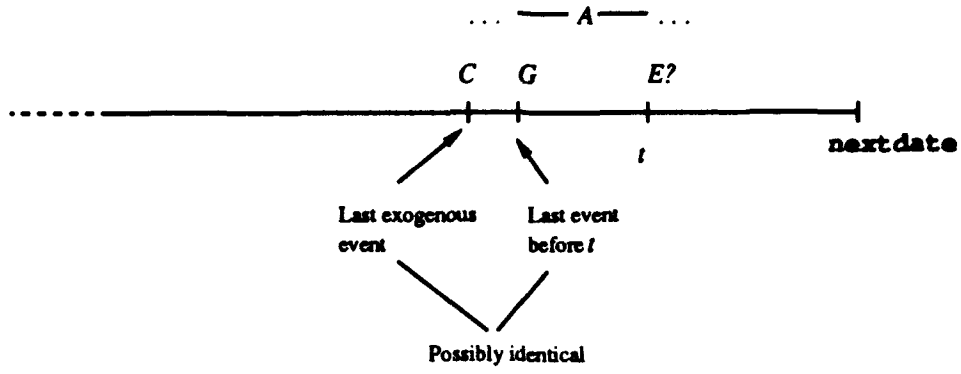


Figure 2: Generating Autonomous Event $E \downarrow t$ Given $A \uparrow t$

Proof: **TIMELINE-ADVANCE** adds eventinstants only to the end of a timeline, using **Add-eventinstant**. **Add-event** then runs forward chaining rules, calling **TL-RETRIEVE** on the past of the new timeline (which = the last time point of the old timeline). By Lemma 2, **TL-RETRIEVE** leaves the timeline in the proper state up through that time point. So all I have to do is show that the later machinations of **Add-event** obey clauses 1 through 3 of the lemma. Clause 1 follows because **Add-event** uses **Conclude-from-rule** (Lemma 1). The proof of clause 2 is essentially the same as for clause 3 of Lemma 2, again because of the use of **Conclude-from-rule**. The proof of clause 4 is obtained by inspection of **Check-clipping**, which is essentially the same as **Bring-forward**, *but applies to every element of the* **overlapping-occasions** of the previous time point. QED

Lemma 4 **TIMELINE-ADVANCE** creates new events in accordance with clause 5 of Definition 5.

Proof: For simplicity, we will consider the case where an event is created by exactly one firable rule instance, of the form $A \xrightarrow{d} E$, and let $\lambda_1 = 1/d$. What we want to prove is that

$$P(E \text{ occurs in interval } [t, t+dt) \mid A \downarrow t) = \lambda_1 dt$$

in accordance with clause 5 of Definition 5. In this formula, P stands for probability defined over execution traces of **TIMELINE-ADVANCE** for a given set of exogenous occurrences. There are an infinite number of such traces, and, if we wanted to be technical about it, we would define carefully how to impose a measure on them. However, these technicalities would take us far from the topic of the paper, and would not reveal any unsurprising phenomena, so we will just use common sense.

The only way for **TL-RETRIEVE** to conclude that A is true at t is for it to conclude that A is true after some previous occurrence G, and for no other events to be added to the timeline from then until t. G must follow or be identical to C, the last exogenous event before t.[9] See Figure 2.

Hence, abbreviating "an occurrence of E is placed in interval $[t, t+dt)$" as "$E \downarrow t$"; and "no event is created between t_1 and t_2 as $N(t_1, t_2)$"; and "the set of firable $\xrightarrow{}$' rule instances at t_1 is F," as $F \downarrow G$,

[9] There are execution traces in which an infinite number of events happen after A becomes true (in fact, there are some in which A changes truth value infinitely often just before t), but the set of all such execution traces is of measure 0, and **TL-RETRIEVE** will run forever in such cases, never concluding that A is true at t.

we have

$$P(E\downarrow t \mid A\downarrow t) = P(E\downarrow t \mid \exists G, F(date(C) \le date(G) < t \wedge A\downarrow G \wedge F\downarrow G \wedge N(date(G), t)))$$

Consider an arbitrary occurrence G with $date(C) \le date(G) < t <$ **nextdate**, and consider an arbitrary timeline in which G has just been added, and **Event-rule-instances** has then picked the set F of all firable rule instances (which it will by Lemma 2). Their frequencies λ_i add up to λ. (λ_1 is one of them, of course.) Let $\Delta t =$ **nextdate** $-$ **date**(G), the size of the interval into which events may or not be placed. If the algorithm creates an event, it creates it with date **when**$+date(G)$, and the cumulative distribution of **when** has $P($**when** $+ date(G) \le t) =$

$$\frac{1 - e^{-\lambda(t - date(G))}}{1 - e^{-\lambda \Delta t}}$$

This formula can be verified by finding the cumulative distribution of **when** given that **x** is uniformly distributed (the standard "inversion trick" for generating nonuniform random numbers; see, e.g., Bratley et al. 1987). The probability that the event occurs before t is this number $\times$ the probability that the algorithm creates an event at all, which is $1 - e^{-\lambda \Delta t}$. The resulting cumulative distribution for **when** is

$$P(\textbf{when} + date(G) \le t) = 1 - e^{-\lambda(t - date(G))}$$

We are interested in the probability that t is the point the program picks for the first event to occur. This value is of course zero, unless we express it as a probability density:

$$\begin{aligned}
\lim_{h \to 0} \frac{1 - e^{-\lambda(t - date(G) + h)} - (1 - e^{-\lambda(t - date(G))})}{h} &= \lim_{h \to 0} \frac{e^{-\lambda(t - date(G))} - e^{-\lambda(t - date(G) + h)}}{h} \\
&= \lim_{h \to 0} \frac{1}{h}\left(1 - \frac{e^{-\lambda(t - date(G) + h)}}{e^{-\lambda(t - date(G))}}\right) \\
&= \lim_{h \to 0} \frac{1 - e^{-\lambda h}}{h} \\
&= \lambda
\end{aligned}$$

The probability that our rule fires immediately after t is then $\lambda\, dt \frac{\lambda_1}{\lambda} = \lambda_1 dt$. This value does not depend on the particular timeline or event G, so it is in fact the $P(E\downarrow t \mid \exists G, F \ldots)$, so

$$P(\text{ some occurrence of } e \text{ between } t \text{ and } t + dt \mid A\downarrow t) = dt/d$$

QED

Timelines are created with **START-TIMELINE**:

```
START-TIMELINE()
  Let tl=Create a new timeline with one eventinstant with happenings (START)
    Forward-chain('(START),tl);
    tl
```

We need an entry point to the system that allows a program to ask about an arbitrary point in the past, not just the last time point:

```
PAST-RETRIEVE (query,timepoint,timeline)
  Let <anses,newoccs>=TL-RETRIEVE(query,timepoint,timeline)
    Bring-forward-until-clipped(newoccs,last-time-point(timeline),
                                timepoint,timeline);
    anses
```

PAST-RETRIEVE simply calls **TL-RETRIEVE**, then makes sure any new occasions created get clipped in the right place, using this subroutine:

```
Bring-forward-until-clipped(persisters,endpoint,beginpoint,timeline)
 If null(persisters) or endpoint=beginpoint
    then persisters
    else let persisters =
            Bring-forward-until-clipped(persisters,past(endpoint),
                                        beginpoint,timeline)
         Bring-forward(persisters,endpoint,timeline)
```

This routine works by bringing persisters forward to `past(endpoint)` recursively, then using **Bring-forward** to get them from `past(endpoint)` to `endpoint`.

Theorem 5 If a timeline is created by **START-TIMELINE**, enlarged with **TIMELINE-ADVANCE**, and queried by **TIMELINE-RETRIEVE** and **PAST-RETRIEVE**, then the resulting timeline state will be properly established, properly clipped, and drawn from the distribution specified by Definition 5.

Proof: **START-TIMELINE** creates and initializes a timeline (using **Forward-chain**). Because **Forward-chain** uses **Conclude-from-rule**, it generates initial time points with the correct statistics and tidiness, by an argument similar to that in the proof of Lemma 2. **TIMELINE-ADVANCE** AND **TIMELINE-RETRIEVE** maintain these properties (Lemmas 2, 3 and 4). **PAST-RETRIEVE** introduces nothing novel, so it can be shown to maintain them, too. QED

3 The Actual Implementation

The actual implementation enhances the idealized algorithm in two ways: greater expressiveness and greater efficiency.

3.1 Enhancements to Expressiveness

The implemented system is more expressive in several ways:

- $\_\leftarrow\_$ rules are fully implemented. The only change is that **Establish-after** must call **TL-RETRIEVE** for the antecedent of such a rule, at the same time point rather than its predecessor.
- In a rule consequent, a conjunct can be of the form $\underline{t}A$, meaning that A becomes true and persists for time t.
- In rule antecedents, it's possible to escape to Lisp for special extensions and low-level computations.
- It is also possible to escape to a simple Prolog-style backward-chaining system for timeless goals. Each rule is of the form $B \leftarrow A_1 \wedge A_2 \wedge \ldots$, where each A_i is handled by another Prolog-style rule or a Lisp handler.
- There is an interface to Lisp for setting up the initial time point.

The implementation is Lisp-based, so rules actually have a Lisp-style syntax. $A \wedge B \ldots$ is actually written (**AND** A B ...), as usual. Here is a translation table for the special symbols of the system, where typically "_" is replaced by P and "|" is replaced by E:

$A\_\leftarrow\_B$	(P<-P A B)
$A/E \mid\overset{r}{\rightarrow}\_ B$	(E->P A E r B)
$B \_\overset{r}{\leftarrow}\mid E\backslash A$	(P<-E r B E A)
$A \_\overset{d}{\rightarrow}\mid E$	(P->E A d E)
$A \leftarrow B$	(<- A B)
$\neg A$ (in antecedent)	(**THNOT** A)
$\neg A$ (in consequent)	(**CLIP** A)
$\underline{t}A$	(**PERSIST** t A)
$A \wedge B \ldots$	(**AND** A B ...)
$p(a, b, \ldots)$	(P a b ...)
Variable v	?v

Rules are asserted by the use of the construct (**DEF-FACT-GROUP** *name* —*rules*—). There is currently only one, global, database of temporal rules allowed.

The presence of **PERSIST** complicates **TIMELINE-ADVANCE** somewhat. Each occasion must have an optional "expiration date" when it will cease to be true. Before **Pass-time** can be called, the system must check for occasion expirations during the time-passage interval. If any are found, it must insert an expiration event, then pass the time before the expiration, then pass the time after the expiration.

It is quite valuable to be able to escape to Lisp. Besides being more efficient for doing arithmetic and such, it allows us to transcend the basic formalism in several ways. Here is a list of some of the more useful extensions implemented this way:

- **(THNOT** *p***)**:[10] **TL-RETRIEVE** doesn't actually have to check for this case specially. It just does a general check for Lisp handlers, and **THNOT** has one that tries to prove *p*, hoping to fail.
- **(OR** *A B* ...**)**: Disjunction
- **(EVAL** *exp val***)**: Evaluate *exp* and unify result with *val*.
- **(<** *m n***)**, **(>** *m n***)**, etc.: Inequalities
- **(LISP-PRED** *r* —*args*—**)**: Run Lisp predicate *r* (e.g., **MEMBER**) on the instantiated *args*. Succeeds if predicate returns **#T**.
- **(START-TIME** *p t***)**: Succeeds if *p* is true now and started at dated *t*.
- **(RECENTLY** *p b e***)**: Succeeds if *p* is not true now, but on the last occasion of *p*, it started at date *b* and ended at date *e*.

The presence of _←_ and **EVAL** allows us to express a sort of rule that is missing in the formal theory developed in Section 2 and Appendix A. Suppose we want to express the idea that if a box contains a certain collection of objects, and one is grabbed, then a random object leaves the box. We can do that thus:

```
(P<-E 1.0 (AND (CONTENTS ?BOX ?NEW)
               (CLIP (CONTENTS ?BOX ?L)))
          (GRAB-FROM ?BOX)
          (AND (CONTENTS ?BOX ?L)
               (EVAL (RANDELT ?L) ?OB)
               (EVAL (DELETE ?OB ?L) ?NEW)))
```

This rule states that when **GRAB-FROM** occurs, the contents of the box change from **?L** to **?NEW**, where **?NEW** is **?L** with an object removed. The choice of removed object is made by the Lisp procedure **RANDELT**. In this way we can eliminate a deficiency in the formalism of this paper, which has trouble expressing the idea that an event can have one of several mutually exclusive outcomes. Other formalisms (e.g., that of Thiébaux and Hertzberg 1992) allow for this possibility, but only in the case where the possible outcomes can be laid out as alternative rules; I don't know of any formalism that allows for the kind of choice among alternatives computed when the rule is applied that the rule above exemplifies. Here we achieve that choice only by escaping to Lisp, and thus leaving the formal semantics behind.

There are other applications of this ability to generate random numbers in Lisp. Suppose we want a non-Poisson model of the occurrences of event type E. Suppose **L** is a Lisp function that generates a random interevent interval according to this model. We introduce a proposition S (for "suppress E") such that E occurs immediately whenever S is false:

[10] This negation-as-failure operator was originally named by Sussman et al. in 1971.

```
(DEF-FACT-GROUP NON-POISSON

  (P->E (THNOT S) 0 E)

  (E->P (EVAL (L) ?LIFETIME) (START) 1 (PERSIST ?LIFETIME S))
  (E->P (EVAL (L) ?LIFETIME) E 1 (PERSIST ?LIFETIME S)))
```

The last expressiveness enhancement I discuss in this section is one of the most important. A key desideratum for a temporal-inference system is that it connect smoothly to users' representations for what's true in the present. If a program is projecting the future, then its beliefs about the present have to be the starting point. Picture a timeline whose initial state corresponds to that set of beliefs. One way to make this picture a reality is to require all users to represent the present as a "short timeline" consisting of just one instant. But there are two reasons to reject this idea. First, it could be clumsy and constraining. Users design their representations with various goals of efficiency and expressiveness in mind that have nothing to do with temporal inference, and we want them to feel free to focus on those goals. Second, some of the information about the present may be uncertain. The program might know that there are between 2 and 4 balls in box1. Our strategy for probabilistic projection is to generate definite projections with different probabilities. So if the temporal-inference system generates several projections, we would like it to pick 2 balls in about 1/3 of the projections, 3 balls in about 1/3, and 4 balls in about 1/3. In each case, there is no uncertainty in the initial state of the timeline.

What we need is an interface between the timeline manager and static representation systems. We provide it with a mutant rule of the form

(INITIALIZER A (f —*args*—))

where A is an atomic formula and f is a Lisp function. To compute instances of A true in the initial state, f is called on the given *args*, which are usually just variables bound in A. f is expected to return a list of the form $((p_1\ l_1)\ (p_2\ l_2)\ \ldots)$, where each p_i is an atomic formula and each l_i is a *lifetime*, i.e., a number or #F. An occasion with the given atomic formula will be created starting in the initial state with the given lifetime (#F means "until clipped"). The p_i do not have to be instances of A.

The key feature is that a given f will be called *just once* for a given set of args. f may use random numbers to set the state up.

Here is how we could handle the situation in which there are an unknown number of balls in box1. Suppose that we represent the number of balls in a box by storing a pair "(l h)" on the property list each box's name, where the actual number of balls lies between l and h. Then we could write the following initialization code:

```
(DEF-FACT-GROUP BALL-BOX-INIT
  (INITIALIZER (IN ?X ?B) (FILL-BOX ?B)))
```

```
(DEFUN FILL-BOX (B)
  (LET ((NUMS (GET B 'NUMBALLS)))
    (COND ((NULL NUMS) '())
          (T
           (LET ((L (CAR NUMS)) (H (CADR NUMS)))
              (DO ((N (+ L (RANDOM (+ 1 (- H L))))
                      (- N 1))
                   (RES '()))
                  ((= N 0) RES)
                (LET ((BALLNAME (GENSYM)))
                   (PUSH '((IN ,BALLNAME ,B) #F)
                         RES)   ))))   )))
```

This example is overly simple and contrived, but you get the idea.[11]

I close this section by explaining how the theory of Section 2.1 is really expressed in the system:

```
(DEF-FACT-GROUP EXAMPLE-THEORY

   ;Initially, NUGGET1 is at 10,0, and...
   (E->P (TRUE) (START) 1.0 (LOC NUGGET1 (COORDS 10 0)))

   ;T1000, an enemy robot, is at 0,0
   (E->P (TRUE) (START) 1.0 (LOC T1000 (COORDS 0 0)))

   ;Roughly every 10 sconds, T1000 moves right.
   (P->E (TRUE) 10.0 (MOVE T1000 1 0))

   ;When an agent moves, its location changes.
   (E->P (AND (LOC ?B (COORDS ?X ?Y))
              (EVAL (+ ?X ?DX) ?X1)
              (EVAL (+ ?Y ?DY) ?Y1))
         (MOVE ?B ?DX ?DY)
         1.0
```

[11] One simplication is that in the actual system, an initializer must also return a "justification" for its answer. The justification system is not mature enough to talk about in this paper.

```
      (AND (LOC ?B (COORDS ?X1 ?Y1))
           (CLIP (LOC ?B (COORDS ?X ?Y)) )))

; Whenever T1000 and the nugget are in the same place,
;T1000 grasps the nugget, typically within 1 second.
(P->E (AND (LOC T1000 (COORDS ?X ?Y))
           (LOC NUGGET1 (COORDS ?X ?Y)))
      1.0
      (GRASP T1000 NUGGET1))

; Grasping succeeds with a probability of 80%.
(P<-E 0.8 (AND (HOLDING ?A ?B)
               (CLIP (LOC ?B (COORDS ?X ?Y))))
      (GRASP ?A ?B)
      (AND (LOC ?A (COORDS ?X ?Y))
        (LOC ?B (COORDS ?X ?Y)))))
```

3.2 Optimizations

A key observation about temporal inference is that many deductions are repeated several times. The procedure **Establish-query** runs _←| rules at a time point, but also calls itself recursively to run them at all previous time points, because a query instance that becomes true earlier could persist until now. Hence the unifications in **Establish-after** will be repeated many times. Similarly, **Check-clipping** will be called every time a new event is added, and it will call **Been-clipped** on every occasion in **overlapping-occasion(last-time-point(*timeline*))**, and run through the _←| rules. The same sort of pattern recurs in typical applications of the timeline. A planning algorithm might issue the request **(LOC AGENT ?W)** repeatedly.

Another key observation is that most queries start at the end of a timeline. The reason is that the timeline is built by repeated calls to **TIMELINE-ADVANCE**, which calls **TL-RETRIEVE** on the antecedent of every |→_ rule. Furthermore, most occasions true at the end of the timeline remain true when new events are tacked on, and they have to be copied to the new eventinstant. Many of these queries end up being propagated into the past, but action in the present dominates.

On top of this, the idealized versions of the algorithms give no details as to how the system is to perform all the retrievals of rules and occasions that are called for. The idealized versions also do not handle **PERSIST** literals, _←_ rules, or **INITIALIZERs**.

To meet all these challenges, the implemented algorithm differs from the idealized version in several respects:

1 When using |→_ and _←| rules, **TIMELINE-RETRIEVE** does not bother to check that the annihilation of the consequent is true before the event. In many cases, the check would be redundant, and usually the rule writer will not mind making the check explicit when necessary. Because of this optimization, it is possible for there to be two overlapping occasions with the same proposition, which is formally meaningless. That means that a program that queries the timeline may have to be careful about duplicate answers. However, one thing we do not have to worry about is the possibility that the multiple occasions will trigger later |→_ rules multiply, causing two overlapping occasions to yield four consequences, which then yield eight, and so forth. That's because the first time an occasion is created (by **Conclude-from-rule**) starting at a eventinstant, an entry is made for its proposition in the established-queries table for the eventinstant, so **Conclude-from-rule** will avoid creating a second one starting at the same eventinstant.

2 Occasions are not stored in lists, but discrimination trees (Charniak et al. 1987). I call these *formula trees*. Given a formula with free variables, one can efficiently retrieve from a formula tree all the occasions it contains whose propositions match that formula.

3 Rules are also indexed, albeit somewhat less efficiently. Different kinds of rules are indexed as they will be retrieved, each associated with one or more symbols. |→_ (**E->P**) rules are accessible from the main functor of their events, because they will be used in **Forward-chain** just after an event has been created. _←| (**P<-E**), _←_ (**P<-P**) and ← (**<-**) rules are indexed by each predicate in their consequents. _→| (**P->E**) rules are left unindexed, because they are all tried every time the timeline is extended.

In each case, the system actually stores a pointer from the indexing symbol to a list of all fact-group names that contain a relevant rule. The reason for this is to simplify redefinition of fact groups.

4 A timeline, formerly just a list of eventinstants, acquires several new fields, and now looks like this:

- **FIRST**: A pointer to the time point for the **START** eventinstant.
- **LAST**: A pointer to the last time point (so far).
- **INDEFINITE-PERSISTERS**: A formula tree containing all occasions that overlap with the last eventinstant of the timeline.
- **EARLIEST-EXPIRER**: The date when the next occasion in **INDEFINITE-PERSISTERS** will expire, or **#F** if all of them will persist until clipped.
- **EXPIRERS**: A list of all occasions in **INDEFINITE-PERSISTERS** with definite expiration dates.
- **QUERIES**: A table, indexed by predicate, of all queries that have ever been issued for this timeline.
- **EVENT-RULES**: All the _→| rules, collected once from fact groups when the timeline is initialized.

The **QUERIES** table can be used to "uniquify" queries. That is, when a query comes into the system, the first action is to see if a variant of this query has been handled before. If not, this one is stored in the table. If a variant is found, **TL-RETRIEVE** uses the stored variant instead of the original query. This tactic allows us to use **EQ** to test for whether a query is established, rather than a much more expensive variant test (which needs to happen just once).

The system uses the same table to uniquify occasion propositions, for much the same reason.

5 An eventinstant actually looks like this:

> **DATE**: When it happens

> **TIMESTAMP**: A finer-grained time measure, used by applications. (Two consecutive eventinstants can have the same **DATE**, but not the same **TIMESTAMP**.)

> **HAPPENINGS**: Events that happen here; () if nothing happens.

> **BEGINNERS**: A formula tree containing occasions that start here and are eventually clipped.

> **PERSISTERS**: A formula tree containing occasions that started before this eventinstant and are clipped at some later eventinstant.

> **TEMPS**: A formula tree containg occasions derived from _←_ rules. These occasions do not persist past the next eventinstant.

> **ESTABLISHED**: A list of queries that have been established here. The queries are uniquified — that is, they all appear in the **QUERIES** table for this timeline —, so the system can use **EQ** to search it.

> **CLIPNOTES**: As before, a table of (atomic-formula, boolean) pairs recording whether any occasion with that formula will be clipped at this eventinstant. These formulas are also uniquified, so an **EQ** test can be used to check the clip status of an occasion.

We still have some occasions associated directly with an eventinstant, but only those whose endpoints have been seen. Those still true at the end of the timeline (so far) will be stored in the **INDEFINITE-PERSISTERS** table for the timeline. When a new eventinstant is added, clipped occasions must be moved from this table to the eventinstants they overlap. But most of the table stays the same.

The procedure **Match-against-occasions** now must check four sets of occasions: the **BEGINNERS**, **PERSISTERS**, and **TEMPS** of the eventinstant in question, plus all elements of **INDEFINITE-PERSISTERS** that overlap the eventinstant.

6 As rules are defined, the system keeps a table that associates an event functor with all the predicates whose occasions it can possibly clip by way of **P<-E** rules. E.g., If (**MOVE** ...) can clip only occasions of the form (**LOC** ...), then the clip-table entry for **MOVE** is (**LOC**). This table enables **TIMELINE-ADVANCE** to update the **INDEFINITE-PERSISTERS** formula tree efficiently. Formula trees are discriminated first by predicate, so the only subtrees that need to be touched when an eventinstant (*A* ...) is added to the timeline are those with a predicate in either the clip table for *A* or the clipnotes for the new eventinstant.

7 An occasion actually looks like this:

* **PROPO**: The proposition, uniquified using the **QUERIES** table of the timeline.

* **BEGIN**: The time point where the occasion begins.

* **END**: If the end of the occasion is known, this is it. Otherwise, this will be a list of "chainrecs," each of which represents a _←| that could clip it. See below.

* **EXPIRATION**: If the occasion has a finite lifetime, this is its expiration date. Otherwise, this field is **#F**.

* **JUSTIF**: The "justification" for the occasion, a record of how it came to occur. In this paper, I do not discuss justification structures, but the idea is to set up data dependencies to link events with their effects.

8 To avoid repeating all those unifications, the system uses the new datatype *chainrec* to keep track of a "deduction in progress." For example, whenever an occasion for proposition A is generated, the system finds all P<-E rules that contain a literal unifying with (CLIP A). For each, it makes a chainrec ("chain record") recording the rule, the substitution, and the result of applying the substitution to the event in the rule. It then sets the **END** field of the occasion to be a list of all such chainrecs. **Been-clipped** then need not consult the database, but simply gets this list from the occasion under consideration. If the list is empty, then the occasion will never be clipped by a P<-E rule.

9 The other use of chainrecs is in backward chaining. The **QUERIES** table of a timeline is used to associate, with each query Q, a list of chainrecs for all the P<-E, P<-P, and **INITIALIZER** rules whose consequents contain a literal that unifies with Q. Each such chainrec records the substitution resulting from a successful unification of Q with some literal in the consequent of a rule. **Establish-query** must then try these rules at every eventinstant.

Actually, **Establish-query** can do a lot better than that, because what the system stores with each uniquified query is actually two lists, the "start chainrecs" and the "regular chainrecs." The former list is for **INITIALIZER** rules and rules of the form (P<-E ... (START) ...). There is no point in trying these rules out anywhere except at the beginning of the timeline, and hence no point in running through a deep recursion to get there. These rules are handled specially by jumping to (START) and then working back to the query point using **Bring-forward**. Regular P<-E rules require repeated calls to **Establish-after**. If there aren't any, the system skips this phase.

4 Experimental Results

In a general rule interpreter like this one, it is impossible to make general statements about run times. It's not hard to make up rule sets that require superexponential times to do projections. However, most rule sets are better behaved. They tend to give rise to the same or similar queries repeated at successive time steps, so many attempts to answer them need only scan back through one eventinstant. Most long scans result from queries that go back to the **START** event to transfer information from the current world model to the timeline.

Here is an example of a theory. It is contrived, but illustrative, and not as violent as some:

```
(DEF-FACT-GROUP INITIALIZERS

  (P<-E 1.0 (KISSED 0) (START) (ALWAYS))

  (P<-E 1.0 (HUGGED 0) (START) (ALWAYS))
```

```
   (P<-E 1.0 (LOC ?X ?X) (START) (ALWAYS))

   (INITIALIZER (PROTECTED ?X) (INITIALLY-PROTECTED ?X)))

(DEFUN INITIALLY-PROTECTED (X)
   (COND ((AND (NOT (HASMVARS X))
               (RANDOM-CHOICE 0.5))
          (VALUES (LIST '(#F (PROTECTED ,X)))
                  #'(LAMBDA () '#T)))
         (T '())   ))

(DEF-FACT-GROUP RANDOM-EVENTS

   (P->E (AND (KISSED ?I)
              (EVAL (+ ?I 1) ?I1))
         10.0
         (KISS ?I1))

   (P->E (AND (HUGGED ?I)
              (EVAL (+ ?I 2) ?I2))
         20.0
         (HUG ?I2)))

(DEF-FACT-GROUP KISS-AND-HUG

    (E->P (KISSED ?I)
          (KISS ?J)
          1.0
          (AND (KISSED ?J)
               (CLIP (KISSED ?I))))

    (P<-E 0.8
          (PERSIST 50 (BLUSHING ?I ?J))
          (KISS ?J)
          (AND (HUGGED ?I)
               (KISSED ?I)
               (THNOT (PROTECTED ?I))))

    (E->P (HUGGED ?I)
```

```
        (HUG ?J)
        1.0
        (AND (HUGGED ?J)
             (CLIP (HUGGED ?I)))))

(DEF-FACT-GROUP KICK-AROUND

    (E->P (AND (NEIGHBOR ?L ?N)
               (LOC ?X ?N)
               (EVAL (+ ?N 8) ?NEW))
          (KICK ?L)
          1.0
          (AND (LOC ?X ?NEW)
               (CLIP (LOC ?X ?N))))   )

(DEF-FACT-GROUP NEIGHBOR-DEF

    (<- (NEIGHBOR ?L1 ?L2)
        (EVAL (- ?L1 2) ?L2))

    (<- (NEIGHBOR ?L1 ?L2)
        (EVAL (- ?L1 1) ?L2))

    (<- (NEIGHBOR ?L1 ?L2)
        (EVAL (+ ?L1 1) ?L2))

    (<- (NEIGHBOR ?L1 ?L2)
        (EVAL (+ ?L1 2) ?L2))   )
```

There are two types of autonomous events, (KISS i) and (HUG i). Initially we have (KISSED 0) and (HUGGED 0); every 5 seconds (on the average) a KICK event occurs, which increments (KISSED i) to (KISSED $i+1$). Similarly for HUGGing, which occurs every 10 seconds on the average, but increments (HUGGED i) by two. If (HUGGED i) and (KISSED i) are ever both true when a (KISS j) occurs and (PROTECTED i) is not true, then an occasion of (BLUSHING i j) begins and persists for 50 seconds. Objects are initially PROTECTED with probability 1/2.

We test the performance of the program by generating exogenous-event streams consisting of $N+1$ events of the form (KICK 5), (KICK 15), ..., (KICK $5+10N$). As N increases, the initial state grows to generate more and more occasions of the form (LOC x x). Here is a table of the execution times that result:

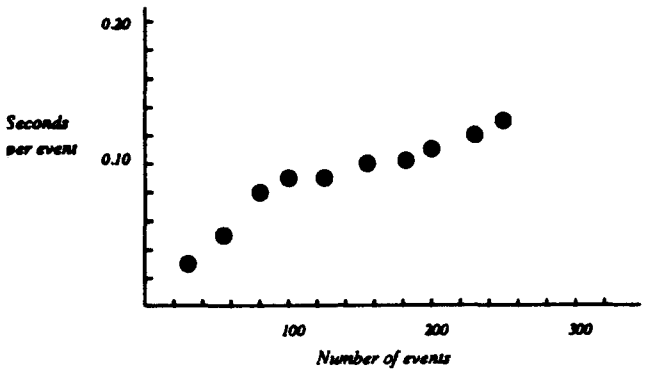


Figure 3: Time Per Event Required to Generate N Events

N	*Run time*	*Number of events*	*Time/event*
10	1.0	28	0.03
20	3.4	54	0.06
30	6.2	81	0.08
40	8.8	101	0.09
50	11.8	128	0.09
60	15.0	154	0.10
70	19.0	183	0.10
80	22.9	202	0.11
90	27.5	230	0.12
100	31.7	249	0.13

These numbers are obtained by running each experiment 10 times and averaging the results. The key column is Time/event. This grows linearly with N, indicating that the total time to generate a timeline with N exogenous events is quadratic in N. (See Figure 3.) It's hard to imagine doing much better than that, given that generating a timeline takes many retrievals in order to run $|\rightarrow\_$ rules, and these retrievals involve a linear component, namely, the time required to check where an occasion added to the beginning of the timeline gets clipped.

5 Conclusions

I have presented a formalism for expressing probabilities of outcomes of events, and shown that it is possible to use rules expressed in this formalism to generate scenarios — "projections" — of what will result from a given event sequence. The formalism has a formal semantics, and I have proven two main results about it:

1. A broad class of rule sets expressed in this formalism yield well defined models.
2. The projection algorithm generates a particularlprojection with the probability specified by the formal semantics.

Furthermore, the algorithm appears to be "efficient." The running times given in Section 4 are not in themselves impressive, but I hope the growth rate is. As the number of events projected grows, the time to generate one more event grows linearly, and slowly. Presumably translating the algorithm from Lisp to C would produce a big absolute speedup, although we have no plans to do that.

This temporal-projection system is now in use in the XFRM planning system being developed at Yale. It is available to interested parties by ftp; please contact mcdermott@cs.yale.edu.

Some enhancements to the system would obviously be worth exploring. The present formalism allows random events to happen at arbitrary real times, but it does not allow for continuous change. For example, we might want to specify that while an agent is moving, its fuel supply depletes continuously with a random slope. It's not hard to see how to alter the implementation to allow for such phenomena, and in fact we have done this on an ad-hoc basis. If a quantity is changing continuously toward some threshold (e.g., an empty fuel tank), then have `Pass-time` check to see if it's gotten there during the time-passage interval being filled in. If it has, then an eventinstant declaring the occurrence of a "reached threshold" event can be added to the timeline. (This is very similar to what happens with expirations of facts with finite lifetimes.)

Unfortunately, as soon as we allow continuous change, the formal analysis will have to be changed significantly. The proofs in the appendix depend crucially on nothing happening between eventinstants, and that assumption will have to be relaxed. Possibly the techniques of Penberthy (1993) will help.

Another class of extensions revolves around allowing rules to make random choices among sets of possibilities. Currently, we can arrange for that only by escaping to Lisp to generate random numbers, and it would be better if the choice was indicated in the rule, and governed by a formal analysis. The problem here is not the formal analysis itself, but figuring out a framework that encompasses all the possibilities. One approach might be to introduce a new sort of rule that allows for labeled random choices at a point. (They have to be labeled so that different rules can refer to different random choices at the same eventinstant.) The choice of whether a rule fires or not is then just a special case.

Acknowledgements: I have learned a lot of probability theory in the course of writing this paper. I thank Dana Angluin, Joe Chang, John Lemmer, and Grigory Matviyenko. Thanks to Matt Ginsberg for suggesting a simpler version of `TIMELINE-ADVANCE` than the one I originally thought of. Michael Beetz and Wenhong Zhu made many helpful comments about an earlier draft of this paper.

A A Rigorous Definition of Timeline Probability

In this appendix, I will go back and make sure that Definition 5 makes sense, by showing that it can be phrased in terms of the limit of a series of finite probability distributions. Throughout this section, I will assume that a set of temporal rules is simple in the sense explained by this definition:

Definition 8 A set of rules is *simple* if and only if

1. There are no $\_ \leftarrow \_$ rules.

2. There is a positive number δ such that if a probability r occurs in a rule instance, $r \notin (0, \delta)$ and $r \notin (1 - \mathit{delta}, 1)$. That is, if r is not 0 or 1, it is in the interval $[\delta, 1 - \delta]$.

3. There is a positive number D such that if a duration d occurs in a $\_ \rightarrow |$ rule, $d < D$.

All of these clauses are for technical convenience. The first clause spares us from having to invest much worry about the static component of the theory. In particular, we can avoid talking about stratification and other tactics for taming recursive backward-chaining rules (Lifschitz 1987). There is, as far as I know, no special technical problem associated with $\_ \leftarrow \_$ rules in a temporal context. The next two clauses do not place a severe limitation on the expressive power of the language, and allow us to draw various useful conclusions. For example, if an infinite set of rule instances have probability $\neq 1$, the product of their probabilities is 0 ($= \lim_{k \rightarrow \infty}(1 - \delta)^k$).

The discrete analog of a world is a finite state series.

Definition 9 A *finite state series* is a mapping from integers to pairs $\langle s, E \rangle$, where s is a world state and E is an event. The domain of a finite state series is a finite set of integers $\{0, 1, \ldots, l - 1\}$, and l is said to be the *length* of the series.

If the ith element of a finite state series is $\langle s, E \rangle$, then that means that at the ith time instant the world is as described by s, and that the next event to occur is E. There is a distinguished event type ϕ that means "nothing happens," and another, Ω, that means "the world ends." Most time points have event ϕ. ϕ is not an element of $\mathcal{Q}$, the set of event types, and no rules refer to it.

Definition 10 An L, μ-*series* for a $\mathcal{T}, \mathcal{C}$ pair, a real number L such that $\mathit{duration}(\mathcal{C}) \leq L$, and a real number $\mu > 0$, where $\mu <$ the smallest interval between two consecutive elements of $\mathcal{C}$, is a finite state series S of length $l = \lceil L/\mu \rceil$, such that the first element of the series has $s = \mathbf{F}$; and if if $\mathcal{C}_j = E \downarrow t$, then $S_{\lfloor t/\mu \rfloor} = \langle s, E \rangle$ for some state s.

Though we do require that every L, μ series have $\mathbf{F}$ as first state, we do not require that it have **START** as first event. It turns out to be technically simpler just to let the first event in $\mathcal{C}$ play the role of initializer, and by convention to have it always be **START**.

In addition to making time discrete, we are going to want to talk about particular kinds of sets of worlds, revolving around this definition:

Definition 11 A *partial state* is the set of all world states that satisfy a (possibly infinite) set of constraints of the following two sorts:

1. Assignments of a truth value (**#T**, **#F**, or $\perp$) to a particular proposition symbol;

2. Assignments of the value **#T** to a non-singleton disjunction of literals.

If an assignment of the first kind is in the set, then the proposition in question is said to be *constrained by* the partial state. A *finite partial state* is one that constrains only a finite set of propositions. An assignment of the second kind is called a *disjunctive constraint*, and it has a secondary role, which will be explained later.

For example, suppose that $\mathcal{P} = \{\mathtt{A}, \mathtt{B}, \mathtt{C}, \mathtt{D}\}$. There is a partial state with four states, all of which specify that A is true and B is false, such that each combination of assignments to C and D is represented. This partial state has no disjunctive constraints, and so we can represent it as a function from propositions to the set $\{\#\mathtt{T}, \#\mathtt{F}, ?, \bot\}$:

$$\begin{aligned} s(\mathtt{A}) &= \#\mathtt{T} \\ s(\mathtt{B}) &= \#\mathtt{F} \\ s(\mathtt{C}) &= ? \\ s(\mathtt{D}) &= ? \end{aligned}$$

Hence, even though partial states are technically sets of world states, I will use the letter s to denote partial states, and will evaluate propositions with respect to them. In such a context, I will take $s(x \wedge ?) = ?$, unless $x = \bot$. I will let S stand for the set of all possible world states, which is the same as the partial state in which all propositions are ?.

Now suppose we add a disjunctive constraint that assigns value #T to $\mathtt{C} \vee \neg\mathtt{D}$. We can think of this as removing from our example partial state the world state that assigns #F to C and #T to D. *However,* I will introduce such constraints only in contexts where the measure of the eliminated world state can be shown independently to be zero. Hence, I will require

Principle V (for vacuity): *The measure assigned to a partial state with disjunctive constraints is exactly the same as the measure assigned to the same state with one or all of the disjunctive constraints eliminated.*

To put it another way: even though a partial state with nonzero probability may include world states that make arbitrary assignments to these propositions, these world states may themselves have probability zero, so that constraining them away with disjunctive constraints does not alter the probability of the partial state.

Hence we can continue to think of partial states as functions from propositions to the set $\{\#\mathtt{T}, \#\mathtt{F}, \bot, ?\}$, and by extension as functions from formulas to this set, but the presence of disjunctive constraints may bring about a situation where a non-? value is assigned to a compound formula even though only ?s are assigned to its constituents. In our modified example, we will still have $s(\mathtt{C}) = s(\mathtt{D}) = ?$, but $s(\neg\mathtt{C} \wedge \mathtt{D}) = \#\mathtt{F}$.

Definition 12 Two partial states s_1 and s_2 *disagree* if and only if there is a proposition symbol A constrained by both such that $s_1(A) \neq s_2(A)$. A partial state s_2 *extends* a partial state s_1 if and only if $s_2(A) = s_1(A)$ whenever $s_1(A) \neq ?$.

Next we define a measure function on L, μ series. (Actually, I will not show it is a measure until after some further lemmas.)

Definition 13 An L, μ-*model* for $\mathcal{T}, \mathcal{C}$ is a pair $\langle \mathcal{W}_\mu, \mathcal{M}_\mu \rangle$, where $\mathcal{W}_\mu$ is the set of all L, μ-series for $\mathcal{T}, \mathcal{C}$ and $\mathcal{M}_\mu$ is a "measure" on $\mathcal{W}_\mu$ that obeys a set of constraints that are discrete analogues of those of Definition 5. In these constraints, we write $E \downarrow i$ to mean the boolean random variable true just for any L, μ-series S in which $S_i = \langle s, E \rangle$ for some s; if A is a proposition symbol we write $A@i$ to mean the boolean random variable true just for any L, μ-series S in which $S_i = \langle s, E \rangle$, where E is some event and $s(A) = \#\mathtt{T}$; and, if s is a partial state, we write $s@i$ to mean the boolean random variable true just for any μ-series in which $S_i = \langle s', E \rangle$ for some $s' \in s$ and some event E. In the following constraints, the letter s is used to refer to partial states; the letter E, to events:

1. For all propositions A in $\mathcal{P}$, $\mathcal{M}_\mu(A@0) = 0$.

2. If $E = \Omega$ then $\mathcal{M}_\mu(s@(i+1) \mid s'@i \wedge E \downarrow i) = 1$ if $s = \bot$ and 0 otherwise. If $E \neq \Omega$, then to compute $\mathcal{M}_\mu(s@(i+1) \mid s'@i \wedge E \downarrow i)$, first let R be a subset of the set of all firable rule instances $\{R_j\}$ of the form $A_j/E \overset{r_j}{\mid\!\!\rightarrow} B_j$ and $B_j \overset{r_j}{\leftarrow\!\!\mid} E \backslash A_j$, where each $s'(A_j \wedge \overline{B_j}) = \#\mathtt{T}$. Let Q be the set of all such rule instances that *might* be firable, that is, the maximal set with $s'(A_j \wedge \overline{B_j}) \in \{\#\mathtt{T}, ?\}$, with $Q \cap R = \emptyset$. Let U be the set of all rule instances whose firability is undefined, that is, those for which $s'(A_j \wedge \overline{B_j}) = \bot$. Treat each B_j as a set of literals. If s can obtained from a subset R^+ of R by adding and deleting consequents of rules in S, that is, if

$$s(b) = \begin{cases} \bot & \text{iff } b \text{ or } \neg b \in B_j \text{ for some } R_j \in U \\ & \text{or } B_j \text{ and } B_k \text{ contain } b \text{ or } \neg b \text{ for distinct } R_j \text{ and } R_k \in R \\ ? & \text{if } s(b) \neq \bot \text{ and } b \text{ or } \neg b \in B_j \text{ for some } R_j \in Q \\ \#\mathtt{F} & \text{if } s(b) \notin \{\bot, ?\} \text{ and } \neg b \in B_j \text{ for exactly one } R_j \in R^+ \\ \#\mathtt{T} & \text{if } s(b) \notin \{\bot, ?\} \text{ and } b \in B_j \text{ for exactly one } R_j \in R^+ \\ s'(b) & \text{otherwise} \end{cases}$$

then

$$\mathcal{M}_\mu(s@(i+1) \mid s'@i \wedge E \downarrow i) = \prod_j \rho_j$$

where $\rho_j = r_j$ if $R_j \in R^+$ and $\rho_j = 1 - r_j$ if $R_j \notin R^+$. If s cannot be derived from such an R^+, then

$$\mathcal{M}_\mu(s@(i+1) \mid s'@i \wedge E \downarrow i) = \begin{cases} \mathcal{M}_\mu(s^+@(i+1) \mid s'@i \wedge E \downarrow i) \\ \quad \text{where } s^+ \text{ is the minimal extension of } s \\ \quad \text{that is derivable from an } R^+ \\ 0 \text{ if there is no extension } s^+ \text{ derivable from an } R^+ \end{cases}$$

3. The computation of $\mathcal{M}_\mu(E \downarrow i \mid s@i)$ splits into two basic cases, depending on whether $E \downarrow i$ is exogenous. If $i = \lfloor date(C_k)/\mu \rfloor$ for some k, then $\mathcal{M}_\mu(E \downarrow i \mid s@i) = 1$ if $E = event(C_k)$, else 0. The endogenous case, when $i \neq \lfloor date(C_k)/\mu \rfloor$ for any k, is more complex. Let R be the set of all rule instances of the form $A_j \overset{d_j}{\rightarrow\!\!\mid} \ldots$ in $\mathcal{T}$ such that $s(A_j) = \#\mathtt{T}$. Let U be the set such that $s(A_j) = \bot$, and let Q be the set such that $s(A_j) = ?$, much as in the previous clause. If U is nonempty, or R is infinite, or $\lambda = \sum_{R_j \in R} 1/d_j > 1/\mu$ then $\mathcal{M}_\mu(E \downarrow i \mid s@i) = 1$ if $E = \Omega$, else 0. If R is finite and Q and U are empty, and $\lambda \leq 1/\mu$, then for every event type E let R_E be the subset of R of the form $\ldots \overset{d_j}{\rightarrow\!\!\mid} E$

$$\lambda_E = \sum_{\left(\ldots \overset{d_j}{\rightarrow\!\!\mid} E\right) \in R_E} \frac{1}{d_j}$$

then

$$\begin{aligned} \mathcal{M}_\mu(E \downarrow i \mid s@i) &= \lambda_E \mu \\ \mathcal{M}_\mu(\phi \downarrow i \mid s@i) &= 1 - \lambda\mu \\ \mathcal{M}_\mu(\Omega \downarrow i \mid s@i) &= 0 \end{aligned}$$

Furthermore, if E_1 and E_2 are distinct event types, it is always the case that $\mathcal{M}_\mu(E_1 \downarrow i \wedge E_2 \downarrow i) = 0$. If U is empty, and $\lambda \leq 1/\mu$, but Q is nonempty, then there is no constraint on $\mathcal{M}_\mu(E \downarrow i \mid s@i)$.

R	R^+	s	Probability
$\emptyset$	$\emptyset$	$\mathtt{B}_1 = \mathtt{B}_2 = \mathtt{C}_1 = ?$	1
$\{1\}$	$\emptyset$	$\mathtt{B}_2 = ?$	$1 - r_1$
	$\{1\}$	$\mathtt{B}_1 = \mathtt{C}_1 = \#\mathtt{T}, \mathtt{B}_2 = ?$	r_1
$\{2\}$	$\emptyset$	$\mathtt{B}_1 = \mathtt{C}_1 = ?$	$1 - r_2$
	$\{2\}$	$\mathtt{B}_1 = \mathtt{C}_1 = ?, \mathtt{B}_2 = \#\mathtt{T}$	r_2
$\{1,2\}$	$\emptyset$	*no change*	$(1 - r_1)(1 - r_2)$
	$\{1\}$	$\mathtt{B}_1 = \mathtt{C}_1 = \#\mathtt{T}$	$r_1(1 - r_2)$
	$\{2\}$	$\mathtt{B}_2 = \#\mathtt{T}$	$(1 - r_1)r_2$
	$\{1,2\}$	$\mathtt{B}_1 = \mathtt{C}_1 = \#\mathtt{T}, \mathtt{B}_2 = \#\mathtt{T}$	$r_1 r_2$

Table 1: Partial States and Their Probabilities

4. If the conditional probability of random variable $E \downarrow i$ or $s@i$ is given by one of the clauses above, then the random variable is conditionally independent of all other random variables for times on or before i. To be precise, if, for random variables α and β, $\mathcal{M}_\mu(\alpha \mid \beta)$ is supplied by Clauses 1, 2, or 3, then for any random variable γ that mentions only times on or before times mentioned in β,

$$\mathcal{M}_\mu(\alpha \mid \beta \wedge \gamma) = \mathcal{M}_\mu(\alpha \mid \beta)$$

Furthermore, if disjunctive constraints are added to partial state $s@i$ in accordance with Principle V, then the conditional probability assigned each such state is the same as that assigned to $s@i$.

An example will make these rules seem clear, I hope. Suppose a theory has just these two rules:

$$\textit{Rule 1:} \quad \mathtt{A}_1/\mathtt{E} \quad \overset{r_1}{\longmapsto} \quad \mathtt{B}_1 \wedge \mathtt{C}_1$$

$$\textit{Rule 2:} \quad \mathtt{A}_2/\mathtt{E} \quad \overset{r_2}{\longmapsto} \quad \mathtt{B}_2$$

and suppose that we are trying to compute $\mathcal{M}_\mu(s@(i+1) \mid s'@i \wedge E \downarrow i)$, where s' is the partial state that assigns #T to $\mathtt{A}_1$ and $\mathtt{A}_2$ and #F to $\mathtt{B}_1$, $\mathtt{B}_2$, and $\mathtt{C}_1$. Table 1 shows the possible outcomes and their probabilities for each choice of R and R^+. The outcomes are specified in terms of the *difference* between the new state and the old. The third column of the table gives all possible partial states that can happen. These are not mutually exclusive. For example, line 2 of the table gives the probability of $\mathtt{A}_1 = \mathtt{A}_2 = \#\mathtt{T}, \mathtt{B}_1 = \mathtt{C}_1 = \#\mathtt{F}, \mathtt{B}_2 = ?$, which is the union of the states described in lines 6 and 8, and has probability $(1 - r_1)(1 - r_2) + (1 - r_1)r_2 = 1 - r_1$. That is, if we don't consider whether a rule fires or not, we get a partial state that ranges over both possibilities.

Definition 14 A theory $\mathcal{T}$ is *consistent* for event sequence C if and only if there exists a δ such that for every $0 < \mu \leq \delta$, no L, μ-model of $\mathcal{T}, C$ has $\mathcal{M}_\mu(\Omega \downarrow i) > 0$ for any i; or $\mathcal{M}_\mu(s@i) > 0$ for any s such that $s(A) = \perp$ for some A. $\mathcal{T}$ is consistent if it is consistent for all C.

Note that the following theory is inconsistent:

$$\mathtt{START} \quad \overset{1}{\longmapsto} \quad \mathtt{P}(1)$$

$$\mathtt{P}(i) \quad \overset{2^{-i}}{\longmapsto} \quad \mathtt{E}$$

$$\mathtt{P}(i)/\mathtt{E} \quad \overset{1}{\longmapsto} \quad \mathtt{P}(i+1)$$

(where each rule involving i stands for all rule instances obtained by substituting an integer > 0 for i). The reason the theory is inconsistent is that the time scale of the second rule decreases exponentially, so that for any event sequence with duration > 1, no matter how small μ gets there will be worlds with $\lambda_{\mathbf{E}} > 1/\mu$, so the event Ω will have nonzero probability.

My strategy is to define the probability of a sequence of partial states in continuous time by taking the limit of the result of adding up the probabilities of all relevant event sequences. We have to be careful about the way we count event sequences, in order to avoid adding up an infinite number of zeros. I start by showing that the probability that an infinite number of events occurs is 0.

Theorem 6 If $\mathcal{T}$ is consistent, then for arbitrary occurrence sequence C and duration $L \geq$ ***duration***(C),

$$\lim_{\mu \to 0} \mathcal{M}_\mu(\lceil L/\mu \rceil \text{ events occur}) = 0$$

where the expression "$\lceil L/\mu \rceil$ events occur" is shorthand for "$\neg\phi \downarrow 0 \wedge \neg\phi \downarrow 1 \wedge \ldots \wedge \neg\phi \downarrow \lceil L/\mu \rceil$-1."

Proof: Since $\mathcal{T}$ is consistent, there is some δ such that for $\mu \leq \delta$, $\lambda\mu \leq 1$, where λ is as defined in clause 3 of Definition 13. Therefore $\lambda\delta \leq 1$ and $\lambda \leq 1/\delta$. Hence the probability that $N = \lceil L/\mu \rceil$ non-ϕ events occur is less than

$$\left(\frac{\mu}{\delta}\right)^{\frac{L}{\mu}}$$

As $\mu \to 0$, this quantity $\to 0$. QED

Theorem 6 means that we can analyze the limit of $\mathcal{M}_\mu$ as $\mu \to 0$ in terms of the probabilities of worlds with a finite number of occurrences. Call each such world a *scenario*. Then we would like to define, for any boolean random variable α,

$$\mathcal{M}(\alpha) = \lim_{\mu \to 0} \sum_{\textit{all scenarios}} \mathcal{M}_\mu([\alpha]_\mu \mid \textit{scenario})\mathcal{M}_\mu(\textit{scenario})$$

where $[\ldots]_\mu$ is defined thus:

Definition 15 The *discretization* $[\alpha]_\mu$ of a formula involving random variables of the form $A \uparrow t$ and $A \downarrow t$ is the formula obtained by making the following substitutions in α:

$$\begin{aligned} E \downarrow t &\longrightarrow E \downarrow \lfloor t/\mu \rfloor \mu \\ A \uparrow t &\longrightarrow A@\lfloor t/\mu \rfloor \mu \\ A \downarrow t &\longrightarrow A@\lfloor t/\mu \rfloor \mu + 1 \end{aligned}$$

Unfortunately, in the limit the probability of most scenarios is zero, because the probability that an endogenous event occurs at a particular time is zero. To sum over all scenarios, we have to group the terms into combinations with non-infinitesimal probabilities. The appropriate groups are what I will call "scenario schemas."

Definition 16 A $\mathcal{T}, C, L$ *scenario schema* is a finite sequence Ξ of triples $\langle \xi_1, \ldots, \xi_k \rangle$, where each ξ_j is of the form $\langle s_j, E_j, t_j \rangle$, written $s_j \downarrow E_j \downarrow t_j$, where s_j is a partial state, E_j is an event type, and t_j is a unique free variable or a date (a real number > 0 and $\leq L$). Those with free variables are called *variable* elements of Ξ; those with dates, *fixed*. The free variables will be bound, and summed and integrated over, in what follows.

The schema is intended to denote the set of all scenarios in which exactly the events $E_j \downarrow t_j$ occur, in the given order, and an element of s_j is true just after the occurrence $E_j \downarrow t_j$. We call this the set of scenarios *described by* Ξ. Ξ must satisfy all of the following:

No two ξ_j have the same t_j;

and for every $E \downarrow t$ in C, there is a $\xi_j \in \Xi$ such that $\xi_j = s_j \downarrow E \downarrow t$;

and the fixed elements of Ξ are ordered by increasing t_j;

and if there is an element $s \downarrow E \downarrow 0$, it is the first element of Ξ; if there is an element $s \downarrow E \downarrow L$, it is the last element.

Definition 17 A scenario schema is *finite* if and only if each of its partial states is finite.

We can discretize scenario schemas:

Definition 18 The *μ-discretization* of a scenario schema Ξ for α is written $[\Xi]_\mu$, and is constructed by transforming every element of Ξ thus:

1. Replace a fixed ξ_j of the form $s_j \downarrow E_j \downarrow t_j$, with $s_j \downarrow E_j \downarrow \lfloor t_j/\mu \rfloor$.
2. Replace a variable ξ_j of the form $s_j \downarrow E_j \downarrow t_j$ with $s_j \downarrow E_j \downarrow i_j$, where i_j is a unique free variable that occurs nowhere else in $[\Xi]_\mu$.

$[\Xi]_\mu$ is said to *describe* the set of L, μ-series in which the E_j happen in the given order, the s_j ensue, and ϕ happens at all time points where no E_j happens.

The probability of a discretized scenario schema cannot be computed directly, but we can define a special case that is easy to compute.

Definition 19 Let j be a position in a schema Ξ ($1 \leq j \leq \mathit{length}(\Xi)$), with triple $s_j \downarrow E_j \downarrow t_j$ occurring at that position. Let the *preceding state* s_p be $\mathbf{F}$ if $j = 1$, and s_{j-1} if $j > 1$. Let $A(R_i)$ be the conjunction of the antecedent of rule instance R_i and the annihilation of its consequent. A T, C, L scenario schema Ξ is *closed at j* with respect to a set of rule instances R if and only if there is a subset R^+ of R such that for every rule instance $R_+ \in R^+$, $s_p(A(R_i)) = \#\mathbf{T}$; and for every rule instance R_- in $R - R^+$, $s_p(A(R_-)) = \#\mathbf{F}$.

Ξ is *closed at j* with respect to proposition symbol P if and only if it is closed with respect to the set of $A/E_j \; |\!\!\rightarrow\_B$ or $B\_\leftarrow\!|\; E_j \backslash A$ rule instances with the literal P or $\neg P$ occurring in B.

Ξ is *closed at j* with respect to E_j if and only if it is closed with respect to the set of all $\_\rightarrow\!|$ rules (not just those for E_j).

Note that if Ξ is a scenario schema for a consistent theory closed at position j with respect to a proposition P, there can be at most one rule instance firable in s_{j-1} that asserts P or $\neg P$; all the others are unfirable in any extension of Ξ.

Definition 20 A scenario schema Ξ is *closed at j*, given that the j'th element of Ξ is $s_j \downarrow E_j \downarrow t_j$, if it is closed at j with respect to E_j, and with respect to every proposition symbol constrained by s_j. A scenario schema is *closed* if it is closed at every j, $1 \leq j \leq \mathit{length}(\Xi)$.

The following theory will provide an example of closedness:

$$\begin{array}{rcl}
\texttt{START} & \overset{0.3}{\mapsto} & \texttt{fragile} \\
\texttt{START} & \overset{0.5}{\mapsto} & \texttt{full} \\
\texttt{drop} & \overset{1}{\mapsto} & \texttt{onfloor} \\
\texttt{fragile/drop} & \overset{0.5}{\mapsto} & \texttt{broken} \\
\texttt{onfloor} \wedge \texttt{broken} \wedge \texttt{full} & \overset{1}{\mapsto} & \texttt{leak} \\
\texttt{leak} & \overset{1}{\mapsto} & \neg\texttt{full}
\end{array}$$

Given $\mathcal{C} = \langle \texttt{START} \downarrow 0, \texttt{drop} \downarrow 1 \rangle$, the following is a closed scenario schema:

$$\begin{aligned}
\Xi_1 = \langle & \{\texttt{fragile}, \texttt{full}\} \downarrow \texttt{START} \downarrow 0, \\
& \{\texttt{broken}, \texttt{onfloor}, \texttt{full}\} \downarrow \texttt{drop} \downarrow 1, \\
& \{\neg\texttt{full}\} \downarrow \texttt{leak} \downarrow t_2 \rangle
\end{aligned} \tag{1}$$

where I have written partial states as sets of literals, all other propositions being unconstrained. In this example, the closedness of the schema depends on every literal except the last occurrence of $\neg\texttt{full}$. That is, if any other partial state were to be deconstrained, the schema would no longer be closed at the next position.

Closed scenario schemas have the useful property that the probability of the set of scenarios represented by a schema is not infinitesimal, as the following theorem will show.

Lemma 7 If $\mathcal{T}$ is a consistent, simple theory, and if $[\Xi]_\mu$ is a discretized closed $\mathcal{T}, \mathcal{C}, L$ scenario for a finite occurrence sequence $\mathcal{C}$, then then for any L, μ-model of $\mathcal{T}, \mathcal{C}$, the measure assigned to the set of L, μ-series described by Ξ, which I'll denote by $\mathcal{M}_\mu([\Xi]_\mu)$, is given by $Q_\mu([\Xi]_\mu, 1)$, where Q_μ is defined by the following recursive construction:

Let $[\Xi]_\mu = \Gamma = \gamma_1, \ldots, \gamma_k$. If j is a position in Γ, let I_j be $i_{\mathit{nextconst}(j)}$, where $\mathit{nextconst}(j)$ is the subscript of the first constant i in Γ that does not precede γ_j, that is, of the next i after γ_j of the form $s \downarrow E \downarrow \mathit{constant}$. (If $\gamma_j = s_j \downarrow E_j \downarrow i_j$ is itself fixed, then $\mathit{nextconst}(j) = j$ and $I_j = i_j$. If γ_j is variable, and j is the last position in Γ, then let $I_j = \lceil L/\mu \rceil$.) Define π_j to be $\mathcal{M}_\mu(s_j @ i + 1 \mid s_{j-1} @ i \wedge E_j \downarrow i)$. This quantity does not depend on i, because s_{j-1} constrains enough propositions to make past times irrelevant. (Take $s_0 = \mathbf{F}$.) Let $\lambda \mid s$ and $\lambda_{E_j} \mid s$ be the λ and λ_{E_j} defined in clause 3 of Definition 13, given partial state s. Once again, we invoke the closedness of Ξ to define these quantities. If $E_j = \phi$, then $\lambda_{E_j} \mid s = (\frac{1}{\mu} - \lambda \mid s)$.

Finally, we can give the definition of Q_μ. The base case is $Q_\mu(\langle\rangle, j) = (1 - \lambda \mid s_{j-1})\mu)^{\lceil L/\mu \rceil - i_{j-1} - 1}$. (Take $i_0 = -1$.)

The recursive case is where $\Gamma = \gamma_j, \Gamma'$. Then

$$Q_\mu(\Gamma, j) = \begin{cases}
(1 - (\lambda \mid s_{j-1})\mu)^{I_j - i_{j-1} - 1}\, \pi_j Q'_\mu \\
\qquad \ldots \text{if } \gamma_j = s_j \downarrow E_j \downarrow \lfloor t_j/\mu \rfloor \text{ for some } E_j \downarrow t_j \in \mathcal{C} \\
(1 - (\lambda \mid s_{j-1})\mu)^{I_j - i_{j-1} - 1} (\lambda_{E_j} \mid s_{j-1})\mu\, \pi_j Q'_\mu \\
\qquad \ldots \text{if } \xi_j \text{ is fixed and not derived from } \mathcal{C} \\
\displaystyle\sum_{i_j = i_{j-1}+1}^{I_j - 1} (1 - (\lambda \mid s_{j-1})\mu)^{i_j - i_{j-1} - 1} (\lambda_{E_j} \mid s_{j-1})\mu\, \pi_j Q'_\mu \\
\qquad \text{otherwise}
\end{cases}$$

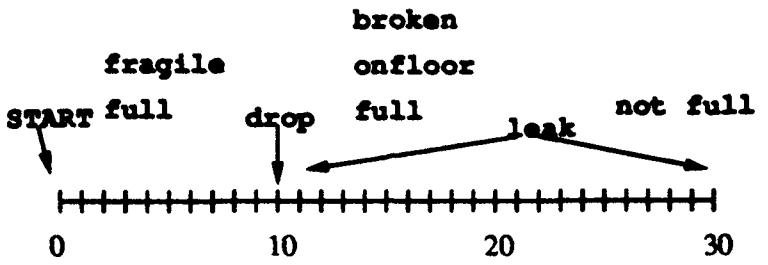


Figure 4: A Discretized Closed Scenario Schema

where $Q'_\mu = Q_\mu(\Gamma', j+1)$, and where i_{j-1} is understood as a variable bound in the $\sum$ for $Q_\mu(\ldots, j-1)$ if γ_{j-1} is variable and $j > 1$, and as the constant i_{j-1} if γ_{j-1} is fixed or $j = 1$.

Proof: This formula may look forbidding, but it's basically just repeated application of Definition 13. Each of the three clauses breaks into four parts:

$$\begin{aligned} &Q_\mu(\Gamma, j) = \\ &\quad \mathcal{M}_\mu(\textit{nothing happens from } i_{j-1}+1 \text{ to } i_j) \times \mathcal{M}_\mu(E_j \textit{ happens at } i_j) \\ &\quad \times \mathcal{M}_\mu(s_j \textit{ becomes true at } i_j + 1) \times \mathcal{M}_\mu(\textit{the rest of } \Gamma \textit{ happens}) \end{aligned}$$

The probability of nothing (i.e., ϕ) happening for l time steps is $(1-\lambda\mu)^l$, where l is the number of steps between E_{j-1} and E_j. In the first two cases, $l = I_j - i_{j-1} - 1$. In the last case, E_j can go anywhere between i_{j-1} and I_j, so we have a $\sum$ that binds i_j to every possible location for E_j. The probability that E_j happens after time i_j is $(\lambda_{E_j} \mid s_{j-1})\mu$, unless E_j is drawn from $\mathcal{C}$, when it's 1. The probability that s_j becomes true after i_j is π_j. The probability that the rest of Γ happens is Q'_μ. In the base case, $Q_\mu(\langle\rangle, j)$ is the probability that nothing happens after the $j-1$'st event, which is just $(\lambda_\phi \mid s_{j-1})^{\lceil L/\mu \rceil - i_{j-1}}$. QED

An example will clarify Lemma 7. Consider the closed schema Ξ_1, discretized with $\mu = 0.1$ (take $L = 3$):

$$\begin{aligned} \Gamma_1 = [\Xi_1]_{0.1} = \langle &\{\texttt{fragile}, \texttt{full}\} \downarrow \texttt{START} \downarrow 0, \\ &\{\texttt{broken}, \texttt{onfloor}, \texttt{full}\} \downarrow \texttt{drop} \downarrow 10, \\ &\{\neg\texttt{full}\} \downarrow \texttt{leak} \downarrow i_2 \rangle \end{aligned}$$

and let $s_1 = \{\texttt{fragile}, \texttt{full}\}, s_2 = \{\texttt{broken}, \texttt{onfloor}, \texttt{full}\}, s_3 = \{\neg\texttt{full}\}$. This schema is illustrated in Figure 4. The I's defined in Lemma 7 are defined thus: $I_1 = 0, I_2 = 10, I_3 = 30$. Abbreviate $1 - (\lambda \mid s_i)\mu$, the probability that nothing happens in one time step where s_i is true, as ρ_i. $\rho_1 = 1, \rho_2 = 1 - 1 \times \mu = 0.9, \rho_3 = 1$. Then

$$\begin{aligned} &\mathcal{M}_\mu(\Gamma_1) \\ &= \mathcal{M}_\mu(s_1@1 \mid \texttt{START} \downarrow 0)\rho_1^9 \times \mathcal{M}_\mu(s_2@11 \mid \texttt{drop} \downarrow 10 \wedge s_1@11) \\ &\quad \times \sum_{i_2=11}^{29} \rho_2^{i_2-11}(\lambda_{\texttt{leak}} \mid s_2) \times \mathcal{M}_\mu(s_3@i_2+1 \mid \texttt{leak} \downarrow i_2)\rho_3^{30-i_2-1} \\ &= 0.15 \times 1 \times 0.5 \times \sum_{i_2=11}^{29} (0.9)^{i_2-11} \times 0.1 \times 1 \\ &= 0.15 \times 0.5 \times 0.86 \\ &= 0.065 \end{aligned}$$

Lemma 7 does not specify a probability for Ξ_1', the schema obtained from Ξ_1 by deconstraining **fragile** after **START**. But intuitively, that probability should be the sum of the probability of Ξ_1 plus the probability of the schema obtained by setting **fragile** to **#F** after **START**. The theorem I will prove next formalizes this intuition, by showing that any set of worlds described by a schema can be reduced to a set described by closed scenario schemas.

Definition 21 A scenario schema Ξ' *extends* scenario schema Ξ if they have the same E_j's and t_j's, and each partial state s_j' in Ξ' extends the corresponding s_j from Ξ.

Definition 22 The *closure* of a scenario schema Ξ is a maximal set of closed minimal extensions Ξ' of Ξ (that is, each Ξ' extends the states of Ξ just far enough to make the resulting schema closed), such that for μ sufficiently small, each $[\Xi']_\mu$ has nonzero probability as specified in Lemma 7.

The following lemma will be useful:

Lemma 8 Let Ξ be a $\mathcal{T}, \mathcal{C}, L$-scenario schema of length I for a consistent, simple theory $\mathcal{T}$, and let S be an infinite set of rule instances such that for each rule instance R in S there is a closed schema Ξ' that extends Ξ and has nonzero probability, and in which R is firable in s_I' (where $\xi_j' = s_I' \downarrow \ldots$). ($S$ is either a set of $\_\rightarrow|$ rule instances, in which case they refer to the probability of various candidate E_{I+1}s, or a set of $\_\leftarrow|$ and $|\rightarrow\_$ rule instances, in which case they must all mention the same E_{I+1}.)

Then there is an arbitarily large finite subset of S such that there is a closed extension of Ξ in which that subset is firable.

Proof: Because there are an infinite number of extensions of Ξ, and each extension mentions only a finite series of events, there must be a partial state sequence $s_0', \ldots, s_k'$ such that an infinite number of Ξ's contain this sequence, and there are an infinite number of distinct extensions of the form $s_0', \ldots, s_k', s_{k+1}', \ldots, s_I'$. (We take s_0 to be $\mathbf{F}$, the initial state with every proposition false.) In other words, if you view the extensions as a tree of schemas, there must be a node of infinite degree (by König's lemma). There may be more than one such partial-state sequence, but we can let k be the position of one of them. Of course, k could be 0, meaning that the extensions branch infinitely at the outset. Label the alternative continuations from that point $s_{k+1}'^{[1]}, s_{k+1}'^{[2]}, \ldots$, and for each $R_\nu \in S$, let $s_I'^{[\nu]}$ be the partial state in which R_ν is a firable rule instance. See Figure 5.

Consider a particular $s_{k+1}'^{[\nu]}$. In $s_{k+1}'^{[\nu]}$ there is a set of propositions constrained to lie in $\{\#\mathbf{T}, \#\mathbf{F}\}$ such that not every other $s_{k+1}'^{[\kappa]}$ constrains them the same way, and whose truth values are such as to guarantee that rule instance R_ν in S is firable with nonzero probability in the corresponding s_I'. Call this the *support for* R_ν in $s_{k+1}'^{[\nu]}$, and give it the name T_ν. This set is finite, because a given $s_{k+1}'^{[\nu]}$ differs from a given $s_{k+1}'^{[\kappa]}$ only in which of a finite set of rules fired in one as opposed to the other. It is possible that an infinite subset must be constrained to fire or not fire to bring about $s_{k+1}'^{[\nu]}$, but all but a finite number must have $r_j = 1$, or the branch containing $s_{k+1}'^{[\nu]}$ would have zero probability. (This is where we use the fact that the theory is simple.) In such a case, all the rule instances with $r_j = 1$ would fire in the generation of *each* alternative $s_k'^{[\kappa]}$, and would not be included in T_ν, which is the set that are specific to $s_{k+1}'^{[\nu]}$.

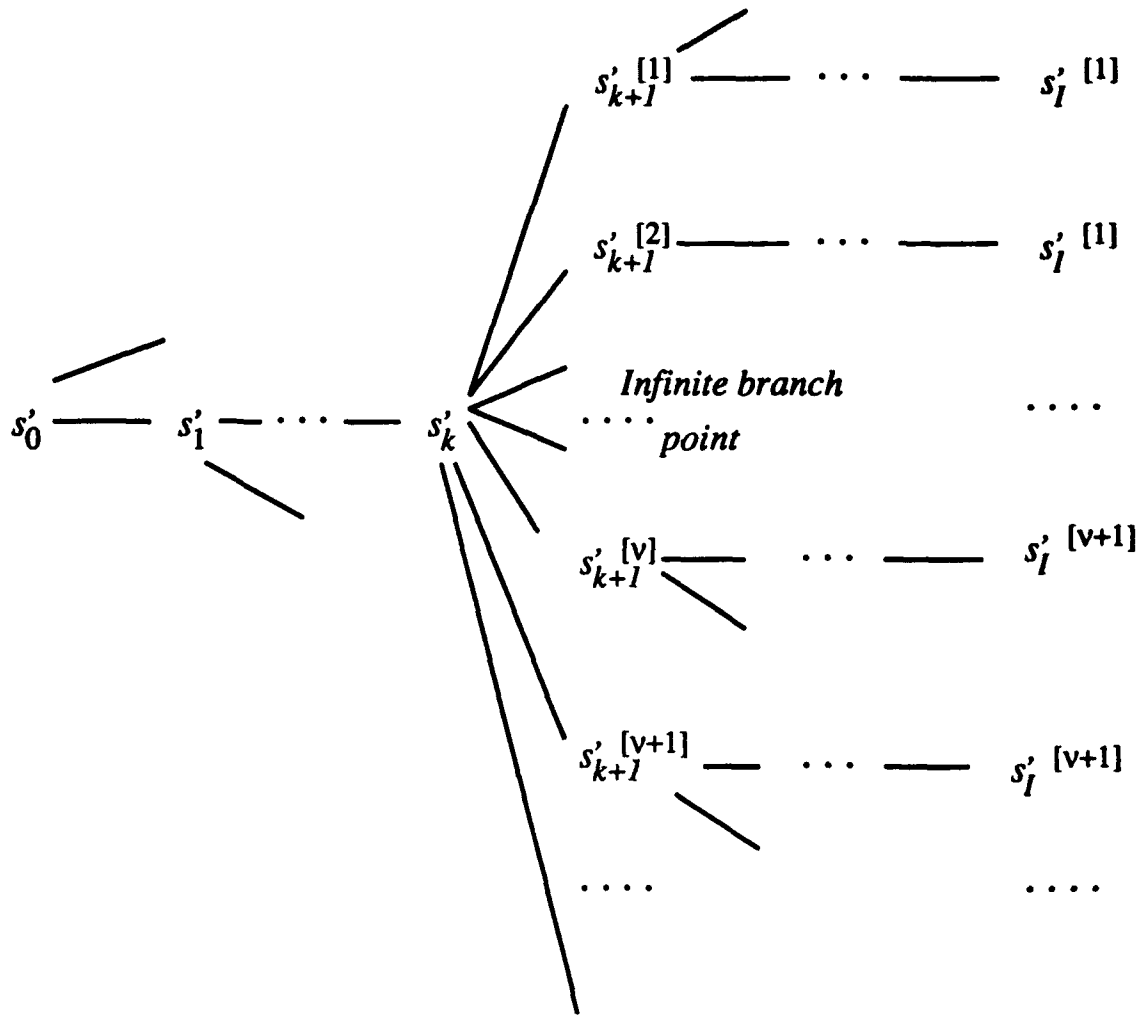


Figure 5: Schemas Arranged as a Tree with Infinite Branch Highlighted

Choose a particular rule instance $\in S$, and call its support T_1. Because T_1 is distinguished from its siblings only by the action of a finite set of rule instances, the presence of T_1 in $s'^{[1]}_{k+1}$ is compatible with all but a finite number of sibling T_ν's. So we can pick an R_2 whose support T_2 is made to appear by the action of a different set of rule instances, and generate a partial state $s'^{[1,2]}_{k+1}$ in which both T_1 and T_2 are present. (This new partial state might or might not be among the s'_{k+1}'s that we already have.) Getting T_1 and T_2 still leaves an infinite number of T_ν's achievable, so we can repeat this process indefinitely, yielding $s'^{[1,2,\ldots,N]}_{k+1}$ for any N, and a scenario schema in which N rules are firable in some successor of $s'^{[1,2,\ldots,N]}_I$. QED

Theorem 9 If Ξ is a finite $\mathcal{T},\mathcal{C}$-scenario schema for a consistent, simple theory $\mathcal{T}$, then Ξ has a unique closure, and that closure is finite.

Proof: The proof is by induction on the length of the schema. A schema of length zero is already closed. Assume the theorem is true for schemas of length I or less, and consider a schema Ξ of length $I+1$,

which ends with $s_{I+1} \downarrow E_{I+1} \downarrow t_{I+1}$. It will be convenient to have Ξ closed at all positions through I, so if it isn't, remove the $I+1$'st element of Ξ, close the shortened schema, and then tack the $I+1$'s element back on to each element of the result. Call this operation *closing the prefix* of Ξ. The result of closing the prefix is, by the induction hypothesis, finite. If it is empty, then the prefix does not describe any timelines, and therefore neither does Ξ, so we are done.

Otherwise, perform the following sequence of operations on each element of the closure. For simplicity, I will use the letter Ξ again for such an element, because all we have done is close Ξ at positions through I. If Ξ is not closed at $I+1$ with respect to every P constrained by s_{I+1}, pick a P for which it is not closed. Find the set S of all rule instances of the form $\ldots\_\leftarrow| E_{I+1}\backslash\ldots$ or $\ldots/E_{I+1} |\rightarrow\_\ldots$ that mention P in their consequents. Pick one rule instance, and let $A_1 \wedge \ldots \wedge A_n$ be the conjunction of its antecedent and the annihilation of its consequent. If possible, create an extension Ξ' of Ξ with $s'_I(A_i) = \#\mathbf{T}$ for each literal A_i. (If $I = 0$, Ξ' will $= \Xi$ unless one of the constraints would require some proposition to have value $\#\mathbf{T}$, in which case Ξ has no extension.) If Ξ' exists, use the induction hypothesis to close its prefix, once again splitting into a finite number of schemas that are all closed through I. We then repeat the process for a set of n candidate extensions, in each of which $s'_I(A_i) = \#\mathbf{F}$ for some $i \in [1, n]$. We have now generated $n+1$ closures of extensions of Ξ. Each schema in any closure is closed at $I+1$ with respect to P. Those in the first closure all have the rule firable; those in the remaining closures, unfirable. See Figure 6(a).

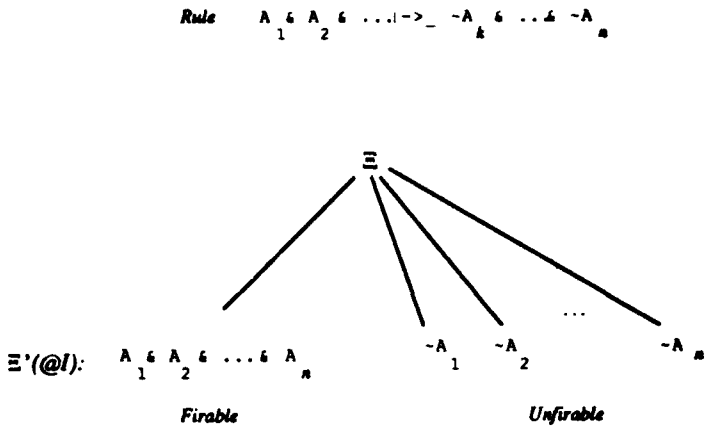
If there is no way to make the first rule firable, then either Ξ' will not exist, or its closure will be empty. In this case, take Ξ' instead to be the schema obtained by taking the prefix of Ξ up to position I, and adding a constraint $s'_I(\neg A_1 \vee \ldots \vee \neg A_n) = \#\mathbf{T}$ (Figure 6(b)). Adding this constraint is in accordance with Principle V because it does not change the probability of s'_I or Ξ', since we have already shown that there is no extension of Ξ that makes the disjunction false. Hence it leaves Ξ' closed through I.

Take all the extensions generated for the first rule instance (just one if the rule cannot fire), and for each repeat the process for the next rule instance that mentions P in its consequent. Continue in this fashion until all rules have been processed. The result is a tree of extensions of Ξ whose leaves are all closed at $I+1$ with respect to P. (Here and for the rest of the proof, if a branch is infinite, define the "leaf" at the end of it to be the schema obtained by taking the intersection of all the partial states along it.) See Figure 7.

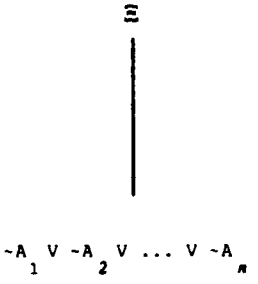
All these extensions are obtained by making truth-value assignments to propositions at positions before $I+1$, so if Ξ' is a "leaf" extension for the tree for P, it will be closed at position $I+1$ with respect to all the propositions Ξ was closed with respect to, plus P. If there are still propositions it is not closed with respect to at position $I+1$, repeat the process with each of them.

When we have generated a bunch of extensions that are closed at $I+1$ with respect to every proposition, we do it again with all the $\_\rightarrow|$ rules.

When this process is complete, we will have produced a big tree of extensions of Ξ. This tree must have a finite number of branches, even if the number of rule instances is infinite, as can be seen by supposing that it has an infinite set of branches. Because Ξ is finite, there are only a finite number of propositions constrained in s_{I+1}. Each of these, by the induction hypothesis, gives rise to only a finite closure at times before $I+1$. So either there is a proposition P for which there are an infinite number of extensions, in each of which a different rule asserting P or $\neg P$ is firable; or there are an infinite number of extensions, in each of which a different $\_\rightarrow|$ rule is firable. In the first case, Lemma 8 says that in some extension of Ξ more than one rule asserting P or $\neg P$ is firable, which is impossible if $\mathcal{T}$ is consistent. In the second case, Lemma 8 says that in some extension of Ξ an arbitrarily large number of $\_\rightarrow|$ rules are firable, and hence λ, as defined in Definition 13, is $> 1/\mu$, no matter how small we take μ, which is also impossible if $\mathcal{T}$ is consistent and simple.



(a) Rule is firable in some extension



(b) Rule is unfirable in any extension

Figure 6: Extending Closure to Make Rule Firable or Unfirable

This construction defines a finite closure for Ξ. It remains to show that it is unique. At each step a minimal extension was made to determine the firability of a particular rule. Suppose the rules had been taken in another order. Any two orderings differ by only a finite number of transpositions of adjacent rules, so suppose R_1 was considered just before R_2, and picture considering R_2 before R_1 instead. If consideration of R_1 does not result in a change to Ξ', the extension to Ξ built so far (that is, if it is not firable in any extension of Ξ'), then clearly we could consider R_2 first and R_1 still wouldn't be firable. If R_1 does generate one or more nontrivial extensions, then first consider the one in which R_1 becomes firable. If R_2 is still firable, then obviously we could have switched the order. If R_2 is not firable, but would have been if we considered it first, then there must be a literal P occurring in the antecedent of R_1 such that $\neg P$ occurs in the antecedent of R_2, and P is not constrained in Ξ'. Hence there is an alternative extension of Ξ' in which R_1 is not firable solely because P is constrained to be false. That extension then gets further extended to a Ξ'' in which R_2 is firable. If R_2 had been considered first, Ξ'' would have been generated as an extension of Ξ' directly. QED

All of the preceding lemmas are consequences of Definition 13. However, I have not shown that $\mathcal{M}_\mu$ is actually a probability measure on sets of L, μ-series. To demonstrate that, we first extend the definition of $\mathcal{M}_\mu$:

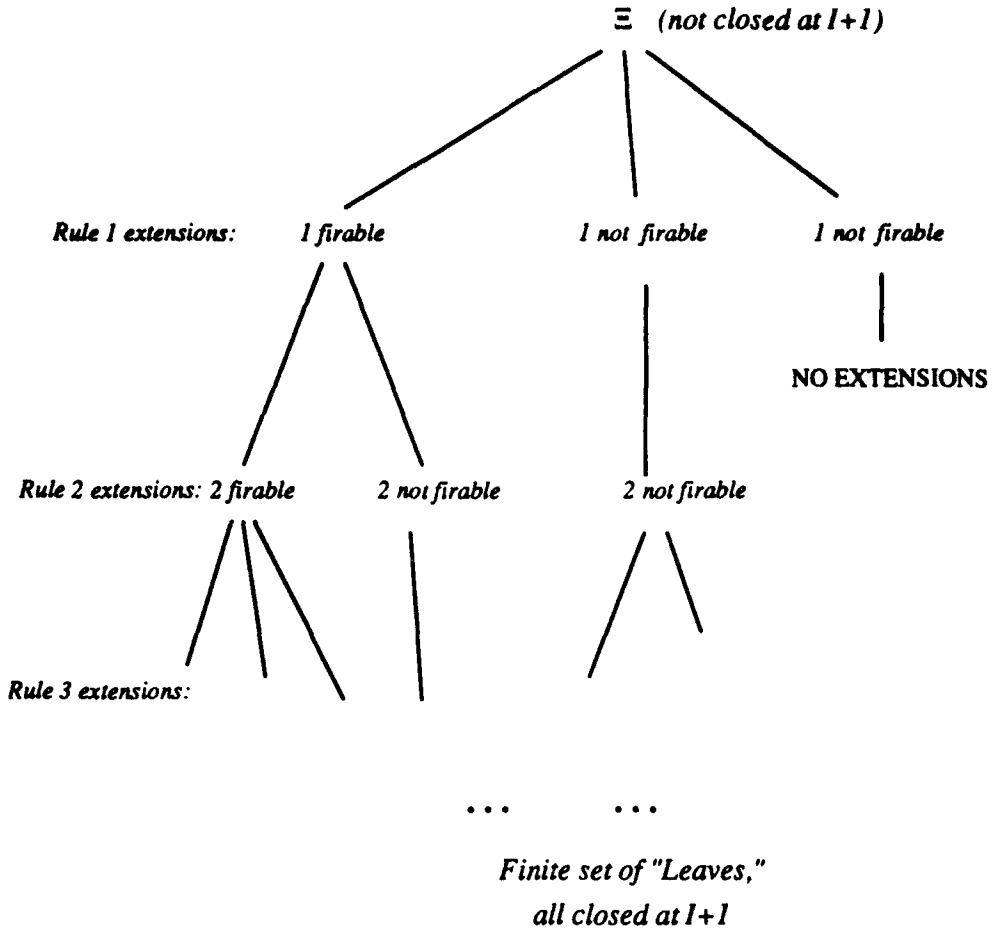


Figure 7: Tree of Extensions, "Leaves" Closed at $I + 1$

Definition 23 If Ξ is a $\mathcal{T}, \mathcal{C}, L$ scenario schema, define the measure of the set of L, μ-series described by Ξ as

$$\mathcal{M}_\mu([\Xi]_\mu) = \sum_{\Xi_c \in CL(\Xi)} \mathcal{M}_\mu([\Xi_c]_\mu)$$

where $CL(\Xi)$ is the closure of Ξ.

Lemma 10 If $\mathcal{T}$ is a consistent theory, and if Ξ is a $\mathcal{T}, \mathcal{C}, L$ scenario schema, P is a proposition symbol, and j is a position in Ξ, with ξ_j, the j'th element of Ξ, $= s_j \downarrow E_j \downarrow t_j$, where s_j does not constrain P, then

$$\mathcal{M}_\mu([\Xi]_\mu) = \mathcal{M}_\mu([\Xi + P \downarrow t_j]_\mu) + \mathcal{M}_\mu([\Xi + \neg P \downarrow t_j]_\mu)$$

where $\Xi + \alpha \downarrow t_j$ means Ξ extended to constrain α to be true in the s_j. (If there is no such extension, take $\mathcal{M}_\mu(\Xi + \alpha \downarrow t_j) = 0$.)

Proof: By induction on j, the position where the extension occurs. The theorem is obvious for $j = 0$, because s_0 constrains every P to be #F. probability of P being true in s_0 is 0. Now assume it's true

through position $j-1$, and consider position $j > 0$. Let Ξ_+ and Ξ_- be the extensions for P and $\neg P$ respectively. If P or $\neg P$ is already true in an element of $CL([\Xi]_\mu)$, then that element will also be an element of $CL([\Xi_+]_\mu$ and $CL([\Xi_-]_\mu$. So consider an element Ξ_c that does not take a position on P at j. The closures of the extensions of Ξ_c for P and $\neg P$ will constrain exactly the same sets of propositions in s_{j-1}, as a glance at Definition 13 will show. For example, in some closed extensions, P will be false at $j-1$ and a rule causing P will be firable with probability r. The Q_μ clause (see Lemma 7) for s_j in the extension for P will differ from the corresponding clause for the original Ξ in having an extra factor of r in computing π_j, while the same clause for the extension to $\neg P$ will have a factor of $1-r$. Similarly, but in the opposite sense, for extensions where P is true at $j-1$ and a rule causing $\neg P$ is firable. In extensions where P is false at $j-1$ and no rule asserting it is firable, we'll give weight 0 to P being true at j, and weight 1 to its being false; and symmetrically in the opposite case. In all cases, wherever there is a closure with weight p for P, there is a closure with weight $1-p$ for $\neg P$. Hence we can combine each of these pairs and rewrite

$$\mathcal{M}_\mu([\Xi + P \downarrow t_j]_\mu) + \mathcal{M}_\mu([\Xi + \neg P \downarrow t_j]_\mu)$$

as a sum of closed extensions of Ξ, each constraining a finite number of propositions at positions j and earlier in all possible ways that make each rule relevant to P firable or not firable. These constraints are mutually exclusive and exhaustive, so, by the induction hypothesis, the sum of Q_μ for these closed extensions is $Q_\mu([\Xi]_\mu)$. QED

Lemma 11 The measure assigned to the set of all L, μ series is 1; that is,

$$\sum_{i=0}^{\lceil L/\mu \rceil} \sum_{\Xi \in S_i} \mathcal{M}_\mu([\Xi]_\mu) = 1$$

where S_i is the set of all $\mathcal{T}, \mathcal{C}, L$ scenario schemas of length i each of whose elements is of the form $S \downarrow E \downarrow t$ for some E.

Proof: Take the closures of each scenario schema, and "ϕ-ify" each of the closed schemas obtained; that is, split each schema into all the possible schemas of length $\nu = \lceil L/\mu \rceil$ obtained by intercalating occurrences of event ϕ. Each such intercalation requires inserting, after a triple $s \downarrow E \downarrow t$, zero or more triples of the form $s \downarrow \phi \downarrow t$, with exactly the same s. The resulting schemas remain closed, because ϕ doesn't change the world state. In addition, inspection of Theorem 7 shows that if a closure element has probability q, then the sum of the probabilities of the schemas derived from it by ϕ-ification is also q.

We now have a collection of closed schemas, all of length ν. We can view them as constituting a tree if we combine prefixes of schemas that agree on a sequence of events and states. Before building this tree, we make sure that if two schemas agree up to a point and then differ, the difference is either that they have different events occurring at that point, or they have disagreeing partial states. We eliminate the case where one schema has a state s_1 that is an extension of the corresponding state s_2 from another schema by invoking Lemma 10 to split the second schema into as many extensions as are necessary to make sure that all either disagree with s_1 or are identical with it. Lemma 10 then says that the sum of the probabilities assigned to the resulting schemas is equal to the schema that contained s_2. In addition, each split schema remains closed, because the distinctions introduced are (a) supported by preceding states; (b) irrelevant to later ones (else the closure would already have taken a stand on the affected propositions).

After these machinations, we can view our collection of closed schemas as a tree of states and events. At the root is the initial state, **F**. Each state has a set of events as children, and each event has a set of states as children. The children of an event E are all nonzero-probability states that can result from

E given that E's parent is the state obtaining just before E; and that can be obtained by combining propositions relevant to the occurrence of a later event. The children of a state are all the events that can occur in the next μ-length time interval if that state obtains just before it (including exogenous events and occurrences of ϕ).

It is now a matter of a simple inductive argument to verify that at each the conditional probabilities of the children given by Definition 13 sum to 1. QED

These two lemmas have a simple corollary.

Theorem 12 $\mathcal{M}_\mu$ is a probability measure

Proof: To be a probability measure, $\mathcal{M}_\mu$ must assign measure 0 to the empty set, 1 to the of all L, μ-series, and must assign $p_1 + p_2$ to $W_1 \cup W_2$ if it assigns p_1 to W_1 and p_2 to W_2. The preceding lemmas allow us to conclude that it does. QED

Now (at last!) we can show what happens as we move to the continuous case.

Theorem 13 If a theory $\mathcal{T}$ is consistent, then for any exogenous occurrence sequence $\mathcal{C}$, and any $\mathcal{T}, \mathcal{C}, L$-scenario schema Ξ with p fixed elements not derived from C,

$$\mathcal{M}(\Xi) \stackrel{\Delta}{=} \lim_{\mu \to 0} \frac{1}{\mu^p} \mathcal{M}_\mu([\Xi]_\mu)$$

exists and equals

$$\sum_{\Xi_c \in CL(\Xi)} Q(\Xi_c, 1)$$

where Q is defined recursively in a way analogously to Q_μ, as follows. Let T_j be $t_{nextconst(j)}$ where $nextconst(j)$ is as defined in Lemma 7. (If $\xi_j = E_j \downarrow t_j$ is fixed, then $T_j = t_j$. If ξ_j is variable, and j is the last position in Ξ, then let $T_j = L$.) Define $\pi_j, \lambda \mid s$, and $\lambda_{E_j} \mid s$ as in Lemma 7.

The base case is now

$$Q(\langle\rangle, j) = e^{-(\lambda_\phi | s_{j-1})(L - t_{j-1})}$$

where we take $s_0 = \mathbf{F}$ and $t_0 = 0$. The recursive case is for $\Xi = \xi_j, \Xi'$, when

$$Q(\Xi, j) = \begin{cases} e^{-(\lambda | s_{j-1})(T_j - t_{j-1})} \pi_j Q' \\ \qquad \ldots \text{if } \xi_j = s_j \downarrow E_j \downarrow t_j \text{ for some } E_j \downarrow t_j \in \mathcal{C} \\ e^{-(\lambda | \theta_{j-1})(T_j - t_{j-1})} (\lambda_{E_j} \mid \theta_{j-1}) \pi_j Q' \\ \qquad \ldots \text{if } \xi_j \text{ is derived from } \alpha \\ \displaystyle\int_{t_j = t_{j-1}}^{T_j} e^{-(\lambda | \theta_{j-1})(t_j - t_{j-1})} (\lambda_{E_j} \mid \theta_{j-1}) \pi_j Q' \, dt_j \\ \qquad \text{otherwise} \end{cases}$$

where $Q' = Q(\Xi', j+1)$, and where t_{j-1} is understood as a variable bound in the $\int$ for $Q(\ldots, j-1)$ if ξ_{j-1} is variable and $j > 1$, and as the constant t_{j-1} if ξ_{j-1} is fixed or $j = 1$.

Proof: First, I prove that for a given closed scenario schema Ξ_c, $\lim_{\mu \to 0} \frac{1}{\mu^p} \mathcal{M}_\mu([\Xi_c]_\mu) = Q(\Xi_c, 1)$. This fact follows from the fact that

$$\lim_{\mu \to 0} (1 - \lambda\mu)^{\lfloor t_2/\mu \rfloor - \lfloor t_1/\mu \rfloor + \nu} = e^{-\lambda(t_2 - t_1)}$$

when ν is a constant. This formula applies directly to the derivation of the clauses of Q's definition from the corresponding clauses of Q_μ. In the second clause, the ^{r}ree $\mu \to 0$, but there are p such factors in the overall definition of $Q(\Xi, 1)$, and they are canceled out by the $1/\mu^p$ factor before the $\sum$. In the third clause, μ becomes a bound dt_j, over which the integration is performed.

So

$$\begin{aligned}
\lim_{\mu \to 0} \frac{1}{\mu^p} \mathcal{M}_\mu([\Xi]_\mu) &= \lim_{\mu \to 0} \frac{1}{\mu^p} \sum_{\Xi_c \in CL(\Xi)} \mathcal{M}_\mu([\Xi_c]_\mu) \\
&= \sum_{\Xi_c \in CL(\Xi)} \lim_{\mu \to 0} \frac{1}{\mu^p} \mathcal{M}_\mu([\Xi_c]_\mu) \\
&= \sum_{\Xi_c \in CL(\Xi)} Q(\Xi_c, 1)
\end{aligned}$$

QED

I can now define the probabilities of propositions in terms of the probabilities of scenario schemas.

Definition 24 If $\mathcal{C}$ is a sequence of exogenous events, and α is a conjunction of random variables of the form described in definition 15, then let $SS_{\mathcal{C}}(\alpha)$ be the set of all scenario schemas $\mathcal{C}$-compatible with α. A scenario schema Ξ is *$\mathcal{C}$-compatible* with α if and only if

1. for each $E \downarrow t$ in $\mathcal{C}$ or α, there is an element $s \downarrow E \downarrow t \in \Xi$ for some s;
2. for each $\neg E' \downarrow t$ in α, there is an element $s \downarrow E \downarrow t \in \Xi$ for some s and some $E \in Q \cup \{\phi\}$, $E \neq E'$;
3. for each t mentioned in α, there is an element $s \downarrow E \downarrow t \in \Xi$ for some s and some $E \in Q \cup \{\phi\}$;
4. for each $A \downarrow t$ in α there is an element $s \downarrow E \downarrow t$ for some $E \in Q$, such that $s(A) = \#\mathbf{T}$;
5. for each $A \uparrow t$ in α there are two successive elements $s_1 \downarrow E_1 \downarrow t_1$, $s_2 \downarrow E_2 \downarrow t_2 \in \Xi$ such that $t_2 = t$ and $s_1(A) = \#\mathbf{T}$ (unless $t = 0$, in which case Ξ is compatible with α only if A is a negated proposition symbol);
6. and for each triple $s \downarrow E \downarrow t$ in Ξ, s is not constrained except as stipulated above.

In general, $SS(\alpha)$ will be infinite, on two counts: First, there can be an infinite number of event sequences that make α true; second, if $A \downarrow t$ or $\neg E' \downarrow t$ occurs in α, and Q is infinite, then $SS(\alpha)$ will have to include an element $s \downarrow E \downarrow t$ for every $E \in Q \neq E'$. However, in the case of infinities of the second kind, most of the schemas will have probability 0, assuming the theory is consistent.

Definition 25 If β is an arbitrary boolean combination of random variables of the form described in definition 15, then express β as a disjunction of mutually exclusive conjunctions of random variables $\alpha_1 \vee \ldots \vee \alpha_k$, and define $\mathcal{M}(\beta)$ to be

$$\sum_{\alpha_i} \sum_{\Xi \in SS(\alpha_i)} \mathcal{M}(\Xi)$$

Now all I have to do is show that the clauses of Definition 5 actually follow from this definition of $\mathcal{M}$.

Theorem 14 Let $\mathcal{T}$ be a consistent, simple theory, $\mathcal{C}$ be an occurrence sequence of length n, and L be a real number $\geq$ ***duration***$(\mathcal{C})$. If $\mathcal{W}$ is the set of worlds obtained as the limit of $\mathcal{W}_\mu$ as $\mu \to 0$, and $\mathcal{M}$ is as defined in Theorem 13, then $\mathcal{M}$ satisfies all the constraints in Definition 5.

Proof: I'll go through the clauses in order.

1. For any proposition $A \in \mathcal{P}$, $\mathcal{M}(A\uparrow 0) = 0$. This is a trivial consequence of Definition 25.

2. If $\mathcal{T}$ contains a rule instance $A/E \stackrel{r}{\vdash\!\!\rightarrow} B$ or a rule instance $B \stackrel{r}{\leftarrow\!\!\vdash} E\backslash A$, then for every date t, require that, for all nonempty conjunctions C of literals from B:

$$\mathcal{M}(C\downarrow t \mid E\downarrow t \wedge A\uparrow t \wedge \overline{B}\uparrow t) = r$$

   To show this clause, start with this fact:

$$\mathcal{M}(C\downarrow t \mid E\downarrow t \wedge A\uparrow t \wedge \overline{B}\uparrow t) = \frac{\mathcal{M}(C\downarrow t \wedge E\downarrow t \wedge A\uparrow t \wedge \overline{B}\uparrow t)}{\mathcal{M}(E\downarrow t \wedge A\uparrow t \wedge \overline{B}\uparrow t)}$$

   Both the numerator and denominator are defined as in Definition 25, and hence each will be a sum of $\mathcal{M}$ values over a set of scenario schemas. Because they mention the same times, the numerator and denominator will sum over the same scenario schemas, except that those for the numerator will have partial states $\downarrow E\downarrow t$ that constrain C to be #T. When the schemas are closed, the two sets retain the same relationship, because any closure that has $A\uparrow t \wedge \overline{B}\uparrow t$ has a rule instance for $C\downarrow t$ that is firable. Hence the denominator will be a sum over a bunch of closed schemas with $\mathcal{S}\downarrow E\downarrow t$, and the numerator will have a similar bunch, but with $s\downarrow E\downarrow t$, where $s(C) = \#\mathtt{T}$. Consulting Definition 13, we see that the ratio of the two partial states' probabilities is r. This ratio will get reflected as the ratios of the corresponding π_j's in the Q for each schema (cf. Lemma 7), and therefore we can pull the r's out:

$$= \frac{r\sum_{\Xi\in CL(S)}\mathcal{M}(\Xi)}{\sum_{\Xi\in CL(S)}\mathcal{M}(\Xi)}$$

   where S = the union of the closures of the schemas for $E\downarrow t \wedge A\uparrow t \wedge \overline{B}\uparrow t$.

3. Suppose B is an atomic formula, and let $R_1, R_2, \ldots$ be the set of all instances of $\vdash\!\!\rightarrow$ or $\leftarrow\!\!\vdash$ rules whose consequents contain B or $\neg B$. (The set may or may not be infinite.) If $R_i = A_i/E_i \stackrel{r_i}{\vdash\!\!\rightarrow} C_i$ or $C_i \stackrel{r_i}{\leftarrow\!\!\vdash} E_i\backslash A_i$, then let

$$D_i = A_i \wedge \overline{C_i}$$

   (The D_i will be identically **false**, in the case where R is empty.) Then

$$\begin{aligned} \mathcal{M}(B\downarrow t \mid B\uparrow t \wedge N) &= 1 \\ \mathcal{M}(B\downarrow t \mid \neg B\uparrow t \wedge N) &= 0 \end{aligned}$$

   where

$$\begin{aligned} N \;=\; & (\neg E_1\downarrow t \vee \neg D_1\uparrow t) \\ & \wedge(\neg E_2\downarrow t \vee \neg D_2\uparrow t) \\ & \ldots \end{aligned}$$

To prove thise clause, we have to rewrite N as a disjunction of mutually exclusive conjunctions. At most one of the E_i can happen, so we can start by trying to express N in the form:

$$\begin{aligned}
&(\neg E_1 \downarrow t \wedge \neg E_2 \downarrow t \wedge \ldots) \\
\vee\ &(E_1 \downarrow t \wedge F_{11}) \\
\vee\ &(E_1 \downarrow t \wedge F_{12}) \\
\vee\ &\ldots \\
\vee\ &(E_2 \downarrow t \wedge F_{21}) \\
\vee\ &\ldots
\end{aligned}$$

where F_{ij} is the j'th conjunction that prevents any rule for E_i from firing. There are duplicates among the E_i, so for a given i we produce the list $F_{i1}, \ldots, F_{il_i}$ by finding conjunctions of literals $\downarrow t$ that (a) mention every proposition found in a rule for E_i; and (b) negate at least one literal in each rule for E_i. The resulting F_{ij} for a given E_i are mutually exclusive, so each conjunct of the disjunction corresponds to a separate scenario schema. That is, either no E_i occurs, or exactly one E_i occurs plus one of the $2^{k_i} - 1$ mutually exclusive ways in which its D_i can be false. For each of these possibilities, create a scenario schema, and find its closure. No rule for any E_i is firable in any element of the closure, so according to Definition 13, the probability is 1 that $B@\lfloor t/\mu \rfloor$ keeps the value it has $@\lfloor t/\mu \rfloor - 1$, and this property will be preserved in the limit.

4. For every time point t such that no occurrence with date t is in $\mathcal{C}$, and for every event type E, let R be the set of all rule instances of the form $\ldots \xrightarrow{d_j} E$ in $\mathcal{T}$, and suppose that S is an arbitrary subset of R, and, letting $r_j = A_j \xrightarrow{d_j} E$, define

$$A = (\bigwedge_{r_j \in S} A_j) \wedge (\bigvee_{r_j \in R-S} \neg A_j$$

and $\lambda_S = \sum_{r_j \in S} 1/d_j$. Then if $\mathcal{M}(A \uparrow t) \neq 0$, require that

$$\mathcal{M}(\text{ some occurrence of } E \text{ between } t \text{ and } t + dt \mid A \uparrow t) = \lambda_S$$

(We have dropped the dt in favor of a convention that $\mathcal{M}$ is a density.)

To prove this clause, express A as a possibly infinite disjunction of possibly infinite mutually exclusive conjunctions of random variables of the form $P \uparrow t$ and $\neg P \uparrow t$. (As in the previous clause, include in each disjunct enough literals that no two disjuncts are consistent). Call the k'th conjunt B_k. By Definition 25, the conditional probability may be expressed as

$$\frac{\sum_k \mathcal{M}(E \downarrow t \wedge B_k \uparrow t)}{\sum_k \mathcal{M}(B_k \uparrow t)}$$

The scenario schemas for the k'th element of the denominator will be the union

$$\bigcup_{E' \in \mathcal{Q} \cup \{\phi\} - \{E\}} \Xi_{k,E'}$$

where $\Xi_{k,E'}$ is the scenario schema for $B_k \uparrow t$ that has the element $\mathcal{S} \downarrow E' \downarrow t$. (Cf. Definition 24.) The closure of $\Xi_{k,E'}$ is exactly the same as that for $\Xi_{k,E}$, except for the switch from E' to E. Hence, when the $\mathcal{M}$ in the denominator is expanded into a sum over the closures of scenario schemas, we can group the terms of the sum into groups that differ only by having a factor $\lambda_{E'}$ in their Q's. These all sum to 1, because the probability of ϕ is defined as $1 - \mu \sum_{E' \in \mathcal{Q}} \lambda_{E'}$ (see Definition 13). Hence the fraction above will be analyzed as the limit of

$$\frac{\lambda_S \mu + \lambda_S \mu + \ldots + \lambda_S \mu}{\mu + \mu + \ldots + \mu}$$

as $\mu \to 0$. In this fraction, the numerator and denominator have the same number of terms, a number that increases as $\mu \to 0$. The limit of the fraction is λ_S.

5. If one of the previous clauses defines a conditional probability $\mathcal{M}(\alpha \mid \beta)$, which mention times t, then α is conditionally independent, given α, of all other random variables mentioning times on or before t. That is, for an arbitrary γ mentioning times on or before t, $\mathcal{M}(\alpha \mid \beta \wedge \gamma) = \mathcal{M}(\alpha \mid \beta)$. This follows from clauses 4, 2, and 3 of Definition 13.

I will end this appendix with a couple of observations about what we have proved. Theorems 9 and 14 imply that a given consistent, simple theory has exactly one model. This situation is quite different from that of first-order logic, where most consistent theories have an infinite number of models. The reason for the difference is the heavy hand of the closed-world assumption (Reiter 1978) applied to the initial state (where all propositions are taken to be #F). Any uncertainty in the system must be modeled in terms of probabilities. So you are allowed to say that the probability is 0.01 that your car has been stolen today, but not to say that you simply don't know whether it has been stolen or not. If we relaxed the closed-world assumption, multiple models would reappear.

Another observation is that the results here apply only when the timeline system is left to its own devices, i.e., when a fixed list of exogenous events is picked in advance, and the question is how the world will react to those events. In practice, the planner uses the projection system this way only to simulate short intervals over which the agent is refraining from action. When a reactive plan is projected, the agent's next actions (i.e., the next exogenous event) is influenced by the state of the timeline just before that action, so the theorems proved here do not apply. For example, in the "Little Nell" problem (McDermott 1982), we wish to project the chances that our agent will be able to protect Nell, who is tied to the tracks, from being crushed by a train. The timeline manager alone might give us a high probability of disaster, even if the agent is allowed to choose an arbitrary series of actions in advance. However, if the agent is allowed to base its actions on, e.g., where it projects it will find Nell, then the probability of a happy ending will be higher.

B References

1 James Allen 1984 Towards a general theory of action and time. *Artificial Intelligence* **23**, 2, pp. 123-154

2 Paul Bratley, Bennet L. Fox, and Linus E. Schrage 1987 *A Guide to Simulation,* second edition. New York: Springer-Verlag

3 Leo Breiman 1969 *Probability and Stochastic Processes.* Boston: Houghton-Mifflin Company

4 David Chapman 1987 Planning for conjunctive goals. *Artificial Intelligence* **32**(3), pp. 333–377

5 Eugene Charniak, Christopher Riesbeck, Drew McDermott and James Meehan 1987 *Artificial Intelligence Programming,* second edition. Lawrence Erlbaum Associates,

6 Keith L. Clark 1978 Negation as failure. In (Gallaire and Minker 1978), pp. 293-322

7 Thomas Dean and Mark Boddy 1988 An analysis of time-dependent planning. *Proc. AAAI* **7**, pp. 49–54

8 Thomas Dean and Keiji Kanazawa 1989 A model for reasoning about persistence and causation. *Computational Intelligence* **5**(3), pp. 142–150

9 Thomas Dean and Drew McDermott 1987 Temporal data base management. *Artificial Intelligence* **32**, no. 1, pp. 1–55

10 William Feller 1970 *An Introduction to Probability Theory and Its Applications, Vol. I.* (Third edition). New York: John Wiley & Sons.

11 H. Gallaire and J. Minker 1978 *Logic and Databases.* Plenum Press

12 Matthew L. Ginsberg (ed.) 1987 *Readings in nonmonotonic reasoning.* Los Altos, CA: Morgan Kaufmann Publishers, Inc.

13 Peter Haddawy 1990 Time, chance, and action. In *Proc. Sixth Conf. on Uncertainty in Artificial Intelligence*, pp. 147–154

14 Steven Hanks 1990 *Projecting plans for uncertain worlds.* Yale Yale Computer Science Department Technical Report 756

15 Steven Hanks 1990 Practical temporal projection. *Proc. AAAI* **8**, pp. 158–163

16 Steven Hanks and Drew McDermott 1994 Modeling a dynamic and uncertain world I: symbolic and probabilistic reasoning about change. To appear, *Artificial Intelligence.*

17 Keiji Kanazawa 1992 *Reasoning about Time and Probability.* Brown University Department of Computer Science Report CS-92-61

18 Vladimir Lifschitz 1987 On the declarative semantics of logic programs with negation. In (Ginsberg 1987).

19 David McAllester and David Rosenblitt 1991 Systematic nonlinear planning. *Proc. AAAI* **9**, pp 634–639

20 Drew McDermott 1982 A temporal logic for reasoning about processes and plans. *Cognitive Science* **6**, pp. 101-155

21 Drew McDermott 1992 Transformational planning of reactive behavior. Yale Computer Science Report 941.

22 Judea Pearl 1988 *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Los Altos, CA: Morgan Kaufmann

23 J. Scott Penberthy 1993 *Planning with Continuous Change.* University of Washington Department of Computer Science & Engineering Report 93-12-01

24 Ray Reiter 1978 On closed world data bases. In Gallaire and Minker 1978.

25 Gerald J. Sussman, Terry Winograd, and Eugene Charniak 1971 *Micro-Planner Reference Manual* MIT AI Laboratory Memo 203A

26 Sylvie Thiébaux and J. Hertzberg 1992 A semi-reactive planner based on a possible models action formalization. In In James Hendler (ed.), *Proc. First Int. Conf. on AI Planning Systems*, San Mateo: Morgan Kaufmann, pp. 228–235

27 Peter van Beek 1992 Reasoning about qualitative temporal information. *Artificial Intelligence* **58**, *1-3*, pp. 297–326